SOFTWARE 2000 INC.
1127 Hetrick Avenue • Arroyo Grande CA 93420 • (805) 489-1977 • Telex 658541 sftwre argd ud

May 12, 1987

Mr. Paul L. Rinaldo, W4RI
The American Radio Relay League, Inc.
225 Main Street
Newington, Connecticut 06111

Dear Paul:

Thank you for your letter of April 22nd in which you invited me to attend the May 23-24 meeting of the ARRL Ad Hoc Committee on Amateur Radio Digital Communications. I regret that I will be unable to attend the meeting. The general release of our NET/ROM networking firmware is taking place later this week, and we expect to be extremely busy processing and filling orders and providing start-up support for several hundred new nodes over the next few weeks. In view of the "vaporware" stigma that quickly attaches itself to software developers who fail to deliver products that live up to their promises, I hope you can understand my reluctance to be away on a trip during such a critical time for NET/ROM.

I am enclosing a copy of the NET/ROM amateur radio version documentation and you are hereby authorized to reproduce it in whole or in part for the members of the Digital Committee to assist in your discussions. It is our intention to provide full disclosure of the protocols and algorithms employed by NET/ROM, and to encourage and assist other software developers who wish to create compatible networking software. As a case in point, we have sent a pair of NET/ROMs to Phil Karn KA9Q and are supporting him in the development of a driver for his TCP/IP code that will allow it to use the NET/ROM layer 3 network service to support TCP/IP traffic. We have even offered to incorporate protocol modifications to assist Phil in his efforts, although it now appears no such modifications may be necessary.

We believe that it is both possible and desirable for TCP/IP and NET/ROM to co-exist as complimentary and compatible standards for higher-level networking in the amateur service. Phil's work addresses the networking "power user" who is willing to dedicate a PC and considerable peripheral hardware to his packet station; NET/ROM addresses the "great unwashed masses" of amateur packeteers, and gives them access to sophisticated higher-level networking capabilities with no additional hardware/software investment beyond their existing AX.25 TNC. Inasmuch as NET/ROM's protocols incorporate strict separation between ISO layers 3 and 4, and provide a "pure datagram" network service at layer 3, it is quite feasible for TCP/IP and perhaps other high-level systems to make full use of NET/ROM's network service as a subnetwork. We therefore expect that both NET/ROM and TCP/IP will be accepted by the worldwide amateur community as twin standards.

The other contenders of which I am aware are COSI and TexNet. The only documentation of the COSI effort of which I am aware is a short "brochure" that contains little or no detail about the protocols and routing algorithms used. However, the implication is that this is a pure virtual-circuit system. If this is so, then COSI would be extremely vulnerable to the failure of nodes and paths, an unacceptable situation in the amateur service in view of the high failure rate that one can reasonably expect. In this regard, Phil Karn and I are in total agreement that "pure datagram" is the only reasonable way to go at ISO layer 3. In addition, I believe that the appropriateness of an X.121 hierarchical numbering scheme for nodes is seriously in doubt for the amateur service; at the very least, it is extremely awkward to use in comparison to NET/ROM's callsigns and mnemonic node names. In sum, from the very little information that has been made available concerning the design of the COSI effort, I have to consider the project ill-conceived.

I have read a considerable amount about the TexNet project. It is chock full of wonderful ideas such as the conference server and alert mode, some of which we hope to make available to NET/ROM users in the near future. I believe that TexNet is hampered by two problems, however. One is its dependence on specialized hardware that is not now in volume production and very well may never reach that stage. The other is poorly designed protocols in which the functions of ISO layers 2, 3, and 4 are not clearly isolated. In sum, I believe that TexNet provides excellent functional concepts that other networkers would do well to emulate, but it should most certainly not be used as a basis for protocol standardization.

I am not certain what role the Digital Committee sees for itself at this point with respect to the various contending efforts aimed at higher-level networking in the amateur service. I sincerely hope that the Committee does not intend to try to "legislate" higher-level networking protocols at this point-in-time when amateur networking is in its infancy. As was the case with the AX.25 and VADCG link-layer protocols, the various network- and transport-layer protocols should be given the opportunity to prove themselves, for better or for worse, in the real world of amateur radio, and amateur packeteers should be given the opportunity to "vote with their TNCs" until one or two clear winners emerge.

Incidentally, Ron Raikes (WA8DED) and I will most certainly be attending the ARRL Networking Conference in August, although I do not yet know whether or not we will be presenting a paper. I look forward to meeting you and other members of the Digital Committee on that occasion.

Best regards,

SOFTWARE 2000, INC.

Michael D. Busch (W6IXU)
President

Enclosure.

S2000
SOFTWARE 2000 INC.

MICHAEL D. BUSCH
President

1127 Hetrick Avenue • Arroyo Grande, California 93420 • U.S.A.
(805) 489-1977 • Telex: 658541 sftwre argd • Cable: TURBODOS

Creators of TurboDOS™

NET/ROM

State-of-the-art networking firmware for the TNC-2

Amateur Radio
Version 1 Documentation
May, 1987

S2000
SOFTWARE 2000 INC.
1127 Hetrick Avenue • Arroyo Grande, California 93420

Notices

Authors

The NET/ROM software was written by Ronald E. Raikes, WA8DED. Concept, design, and documentation by Michael D. Busch, W6IXU.

Copyright & Trademark

This document © 1987 Software 2000, Inc.
NET/ROM software © 1987 Software 2000, Inc.
NET/ROM™ is a trademark of Software 2000, Inc.

The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. *It is against the law to make reproductions of NET/ROM software on ROM, magnetic tape, disk, or any other medium for any purpose whatsoever.*

Warranty & Disclaimer

Software 2000, Inc., warrants that it has the right to license the NET/ROM software, and agrees to replace defective copies of NET/ROM within 90 days of purchase provided the original ROM is returned postpaid to Software 2000, Inc., with a self-addressed stamped envelope. Software 2000, Inc., makes no other warranties, either express or implied, regarding NET/ROM software, its merchantability or its fitness for any particular purpose.

Information in this document is subject to change without notice, and is not a binding commitment on the part of Software 2000, Inc.

For More Information...

Software 2000, Inc.
1127 Hetrick Avenue
Arroyo Grande, California 93420
USA

Orders, Prices, Licenses:
Telephone: (805) 489-1977
Telex: 658541 (sftwre argd ud)
CompuServe: Mike Busch [W6IXU] 76337,727

Technical Support:
Telex: 704444 (sftwre dny ud)
CompuServe: Ron Raikes [WA8DED] 76337,727

(Intentionally left blank.)

Table of Contents

Overview	Introduction	5
	Functional Highlights	6
	A Usage Example	9
Theory of Operation	Some Terminology	11
	Making Connections	13
	Digipeating vs. Store-and-Forward	17
	Firmware Organization	19
	Flow of Information	24
	Error and Flow Control	27
	Deferred Disconnect	27
	Automatic Adaptive Routing	28
	Routing Table Updates	31
	Route Quality Analysis	32
	Station Identification	33
	Digipeating	33
	Control Operator Validation	34
Commands	CONNECT Command	35
	NODES Command	37
	USERS Command	39
	SYSOP Command	40
	IDENT Command	41
	PARMS Command	42
	RESET Command	43
	Command Interpreter Messages	43
Host Interface	ESC-C – Connect	45
	ESC-D – Disconnect	45
	ESC-P – Password String	45
	ESC-T – Transmitter Key-Up Delay	46
	ESC-Y – Enable/Disable Host Connections	46
	Host Interface Messages	46
Installation	Hardware Setup	47
	Parameter Setup	51
	Dual- or Multi-Channel Operation	52
Appendices	Appendix A: Protocols	55
	Appendix B: Routing	61
	Appendix C: Parameters	67
	Appendix D: Operating Notes	73
	Appendix E: Index	79

(Intentionally left blank.)

Overview

Introduction

NET/ROM is a firmware replacement which transforms a standard TAPR TNC-2 (or "clone") into a state-of-the-art network node controller. It is intended for use primarily on hilltops and other wide-coverage digipeater sites. Version 1 is not appropriate for installation at end-user or mailbox stations.

High-level networking

A NET/ROM node provides the normal functions of an ordinary AX.25 digipeater, plus a set of sophisticated higher-level networking capabilities. A user may connect to a nearby NET/ROM node; display a list of other known network nodes; establish a transport-layer circuit to a distant node; and connect to another end-user or mailbox in the vicinity of the distant node. Compared with conventional AX.25 multi-hop digipeating, NET/ROM's true store-and-forward packet switching technology can provide an order-of-magnitude improvement in reliability and throughput, especially over long paths. Routing from the local node to the distant node is handled automatically, and even includes alternate routing to circumvent network outages.

Multi-port operation

NET/ROM supports cross-frequency and cross-band networking without the need for exotic multi-port digipeater hardware. A dual-channel node, for example, is implemented simply by installing NET/ROM in a pair of standard TNC-2s and connecting them together with an RS232 cable. A three- or four-channel node can be created by connecting multiple TNC-2s together, using a simple diode-matrix coupler.

Multi-layer protocols

NET/ROM uses the standard AX.25 link protocol for both node-to-node and node-to-user links at ISO layer two. Higher-level NET/ROM protocols provide for a "pure datagram" network service at layer three, plus a "virtual circuit" transport service at layer four.

Overview

Functional Highlights

Runs on ordinary TNC-2

NET/ROM is unique because it provides state-of-the-art networking capabilities without the need for special hardware. NET/ROM runs on a standard TAPR TNC-2 terminal node controller, or on any of the commercially-available TNC-2 "clones". NET/ROM is distributed in the form of a 27C256 EPROM which simply plugs into the ROM socket of the TNC-2 in place of the standard TAPR firmware ROM.

Store-and-forward packet switching

Instead of multi-hop digipeating, NET/ROM provides true store-and-forward packet switching. The result is significant improvement in throughput and reliability. On long paths (three or more digipeaters), dramatic improvements are possible. For instance, the calculated speed improvement for the Los Angeles/San Francisco route (four digipeaters) assuming typical 1200-baud VHF network conditions is about 800%!

Two-layers of error and flow control

NET/ROM uses the standard AX.25v2 packet radio protocol for links between neighboring nodes, as well as for links from each node to its local users. Normal AX.25v2 error handling is used to assure error-free transmission, and normal AX.25v2 flow control is used to manage network congestion.

In addition, NET/ROM incorporates a transport-layer "sliding window protocol" to provide end-to-end error and flow control for each virtual circuit. End-to-end error control protects against lost, duplicate, or out-of-sequence frames resulting from node failures and dynamic routing changes. End-to-end flow control protects the network against disproportionate loading by any one circuit.

Overview

Functional Highlights (continued)

Deferred disconnect logic

When two stations are connected to one another via NET/ROM and one of the stations disconnects, NET/ROM automatically maintains its connection to the other station until all in-transit information frames have been successfully delivered to that station. NET/ROM disconnects only after all in-transit information has been delivered, or after 15 minutes has elapsed without any "forward progress" in delivering such information.

Multi-channel capability without special hardware

NET/ROM supports multi-frequency operation without the need for exotic multi-port digipeater hardware. A dual-channel node, for example, consists simply of a pair of standard TNC-2s (with NET/ROM in each) connected together with a simple RS232 cable. Configuring a NET/ROM node for three or four channels is almost as easy—a TNC-2 is used for each frequency, and the multiple TNCs are interconnected via their RS232 ports using a simple diode-matrix coupler.

... plus "wormholes"

In addition, it is possible to configure a dual-channel NET/ROM node in which the two TNC-2s are not co-located. Instead of an RS232 cable, the TNC interconnect can employ a dedicated telephone line or even a satellite link. Because NET/ROM uses an asynchronous variant of AX.25 over the interconnect, it is not necessary that it be an error-free connection. This opens up fascinating possibilities, such as the experimental satellite-linked dual-channel node accessible from both Baltimore and San Francisco—affectionately known as "the wormhole"!

Functional Highlights (continued)

Automatic adaptive routing	NET/ROM automatically takes care of the routing of traffic between one node and another. A user needs to specify just the desired destination, not the route. Each node keeps track of the other nodes in the network and the various possible paths that may be used to reach them. If a node or path becomes unusable due to equipment failure or poor propagation, NET/ROM automatically changes to an alternate route (if available) to circumvent the outage. Conversely, when a new node is placed on-line, other nodes automatically incorporate the new node into the network routing structure. Such routing changes are handled dynamically, without disrupting user connections in progress.
Routing table updates	NET/ROM supports three methods of updating its routing information: local, remote, and automatic. Initial routing information may be entered manually by an on-site operator using a local terminal. Routing changes may be made remotely by a control operator over an ordinary packet radio connection—a randomized verification algorithm effectively prevents changes by unauthorized operators. In addition, NET/ROM nodes broadcast routing information to each other on an hourly basis, thereby enabling the network to incorporate new nodes and to bypass outages in real-time without manual intervention.
Mnemonic node identifiers	Each NET/ROM node is identified in two ways: by a valid amateur callsign permanently encoded into each copy of NET/ROM, and by an arbitrary mnemonic node identifier established by the node's control operator. Identifiers may be up to six characters long—three-letter city designators used in aviation (LAX, SFO, DCA, ATL, JAX, etc.) are a possible choice. Mnemonic node identifiers appear in node station identification beacons, and are passed to other nodes during hourly automatic routing broadcasts. The node identifier may be used in lieu of the node callsign in most contexts.

Overview

Functional Highlights (continued)

Easy to learn, easy to use

Despite its internal sophistication and advanced networking capabilities, NET/ROM is exceptionally easy to use. A novice user needs to learn only one command, CONNECT, to establish crosslinks to other nodes or downlinks to other user stations. More sophisticated users may wish to use the NODES command to list the callsigns and identifiers of other network nodes, and the USERS command to find out who else is using the node.

Control operators can use SYSOP to validate their control operator privileges, NODES to make manual changes to the routing table, IDENT to establish a mnemonic node identifier, PARMS to set or display various node parameters, and RESET to warm-start the NET/ROM firmware.

Compatible with existing AX.25 digipeaters

Each NET/ROM node also supports the functions of an ordinary AX.25 digipeater. Users need not make use of the high-level networking functions of NET/ROM unless they want to. Digipeater owners can upgrade their sites to NET/ROM nodes without disrupting users. Multi-channel NET/ROM nodes provide multi-port digipeating as well. Mnemonic node identifiers may be used in lieu of node callsigns when specifying a digipeated path.

A Usage Example

You are KA6PRE ("Packet Radio Enthusiast") located in San Diego, and you want to access the W0RLI bulletin board system in Santa Cruz (near San Francisco). First, you establish an uplink to your local node, using the normal connect command provided by your TNC. To connect, you can use either the node's callsign "W6AMT-4" or its mnemonic identifier "SAN":

```
^C
cmd: CONNECT SAN
*** CONNECTED to SAN
```

NOTE: user input is underlined for clarity.

A Usage Example (continued)

Next, you ask the local node for a list of other NET/ROM nodes for which it has routing information, using the NODES command:

```
NODES
SAN:W6AMT-4} Nodes:
LAX:W6AMT-3     SBA:W6AMT-2     PRB:W6AMT-1
SFO:W6AMT       WWORM:WB6FFC-1  EWORM:WA3YMH-1
ONT:AA6TN-1     LAS:K7WS-1      PHX:WB7BNI-1
```

You know that SFO:W6AMT serves the San Francisco area (it was always the last digipeater you used to reach W0RLI via AX.25), so you ask your local NET/ROM node to connect you there (you can use either W6AMT or SFO):

```
CONNECT SFO
SAN:W6AMT-4} Connected to SFO:W6AMT
```

You are now talking to the San Francisco node, and you could issue another NODES command to see a list of other nodes that it knows how to reach. In this case, you simply want a connection to the W0RLI BBS, so you ask for it:

```
CONNECT W0RLI
SFO:W6AMT} Connected to W0RLI
Hello Fred, welcome to W0RLI BBS at 0532z
KA6PRE-15 de W0RLI>
etc.
```

Note that the downlink from W6AMT to W0RLI has been made using your callsign (KA6PRE), but with the SSID suffix changed from -0 to -15. (More about this later.) At the end of your session with W0RLI, you disconnect your TNC as usual:

```
^C
cmd: DISC
*** DISCONNECTED
```

When you disconnect, NET/ROM automatically takes care of disconnecting your circuit to SFO:W6AMT and your downlink to W0RLI.

Theory of Operation

Some Terminology Following are definitions of some terms that are used in describing the operation of NET/ROM.

"User" Any amateur packet radio station using AX.25 protocol. In the context of this document, a BBS or other automated server is considered a "user".

"Digipeater" A station capable of digitally repeating AX.25 frames as specified in the AX.25 protocol specification. Generally, refers to an unattended, wide-coverage digital repeater, often located on a hilltop.

"Node" A packet radio station utilizing TNC-2 hardware executing NET/ROM firmware. Generally, refers to an unattended, wide-coverage station, often located on a hilltop.

"Dual-channel node" A pair of NET/ROM nodes operating on two different frequencies, and coupled together by means of an RS232 cable.

"Multi-channel node" Three or more NET/ROM nodes operating on different frequencies, and interconnected via their RS232 ports using a diode-matrix coupler.

"Link" An AX.25 connection involving a node at one or both ends. Node-to-node links always use AX.25v2 protocol. User-to-node links use AX.25v2 protocol if the user's TNC supports it, otherwise AX.25v1.

"Uplink" An AX.25 connection between a user and a node, initiated by the user. An uplink is usually a direct connection, but may be digipeated if necessary.

"Downlink" An AX.25 connection between a node and a user, initiated by the node. A downlink is usually a direct connection, but may be digipeated if necessary. A downlink is established by a node at the behest of a user (in response to a CONNECT command).

Some Terminology (continued)

"Crosslink" An AX.25 connection between two adjacent nodes. A crosslink is usually a direct connection, but may be digipeated if necessary. A crosslink between two nodes is initiated by one of the nodes when first establishing a circuit which traverses the route segment between the two nodes. One crosslink can carry any number of circuits, so it is never necessary to have more than one crosslink between any pair of nodes.

"Circuit" A transport-layer connection between two nodes, established by one of the nodes at the behest of a user (in response to a CONNECT command). The two nodes are not necessarily adjacent, and may be quite distant. The circuit is automatically routed through intermediate nodes as necessary, and carried from node to node over crosslinks (which are initiated if not already established).

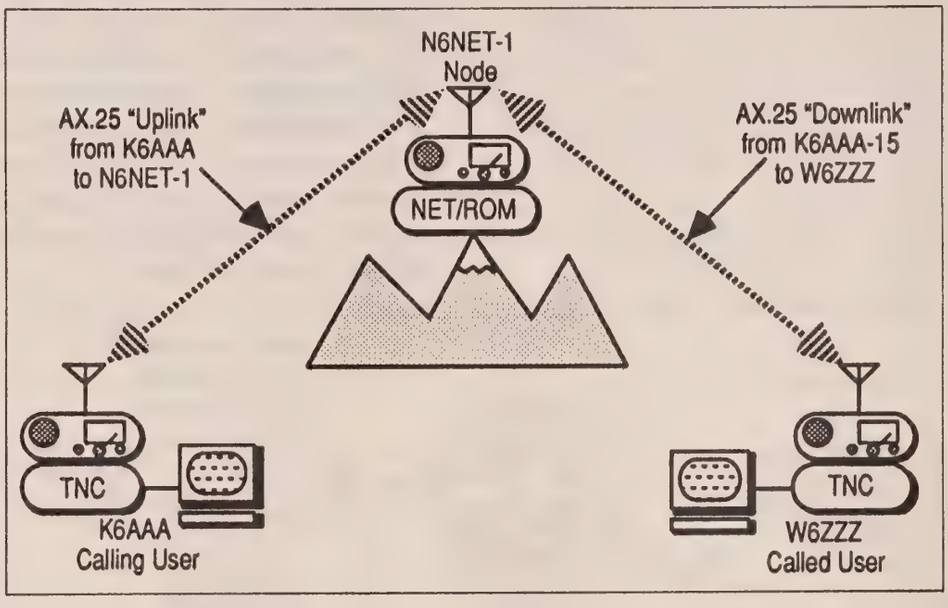
Theory of Operation

Making Connections

Suppose you are a user with access to a local node, and you want to contact another user station who is also within range of the same node. You can, of course, connect to the other station "the old way" by using the node as a digipeater.

Uplink-downlink connection

To take advantage of the store-and-forward capabilities of the node, however, you would use this two-step procedure: (1) connect to the node ("uplink"); then, (2) issue a CONNECT command with the callsign or mnemonic identifier of the other user station ("downlink").



All AX.25 frames include the callsigns of both the originating station and the destination station. When you request a downlink, the node "adopts" your callsign as the originating station (rather than using its own callsign). This is necessary so that the destination station can properly identify you as the connecting user station, and is especially important when the destination is a mailbox, gateway, or other automated server.

Well...that's not *exactly* the way it works! If the node truly did "adopt" your callsign, and if there also happened to be a direct path (however marginal) between your station and the destination station, that station would then be "in range" of two stations using the same callsign. This can create serious confusion, AX.25-protocolwise!

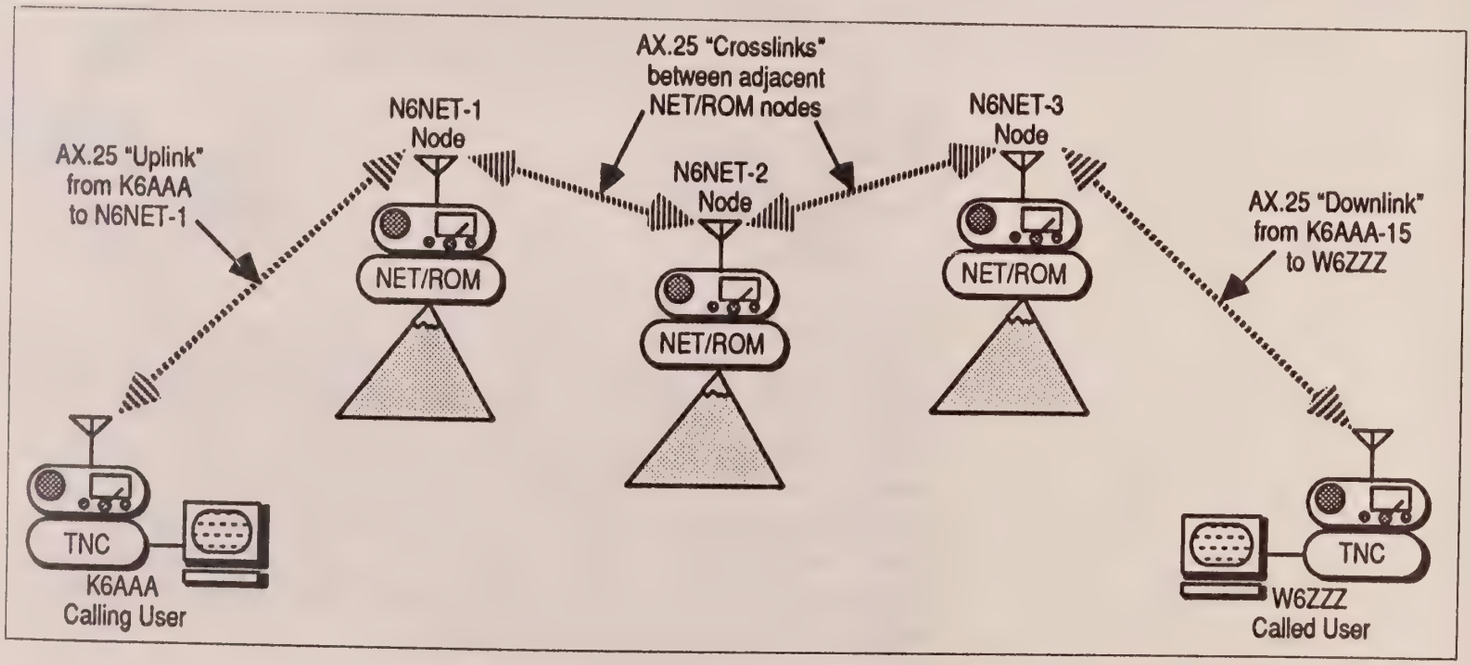
Theory of Operation

Making Connections (continued)

To avoid this problem, the downlinking node "adopts" your basic callsign, but changes the SSID (the "-N" callsign suffix) from *N* to *15-N*. For example, if your callsign is K6AAA, the downlink uses K6AAA-15; if your callsign is W3ABC-2, the downlink uses W3ABC-13; and so forth.

Uplink-crosslink-downlink connection

Now, suppose you want to contact a user station that can't copy your local node, but is in range of another node that serves his area. Once again, you could use "old-style" multi-hop digipeating to reach him (since every node is also a digipeater). To utilize the full store-and-forward capability of the nodes, however, you would use a three-step procedure: (1) connect to your local node; then, (2) issue a CONNECT command with the callsign or mnemonic identifier of the distant node; finally, (3) issue a CONNECT command with the callsign of the other user station.



When you perform step (2) of this procedure, you are asking your local node to create a "circuit" for you between your local node and the distant node. If the two nodes are sufficiently far apart, the circuit may have to pass through several intermediate nodes. In any case, the routing is performed automatically by the node. Your circuit is carried by a series of AX.25 "crosslinks" between pairs of

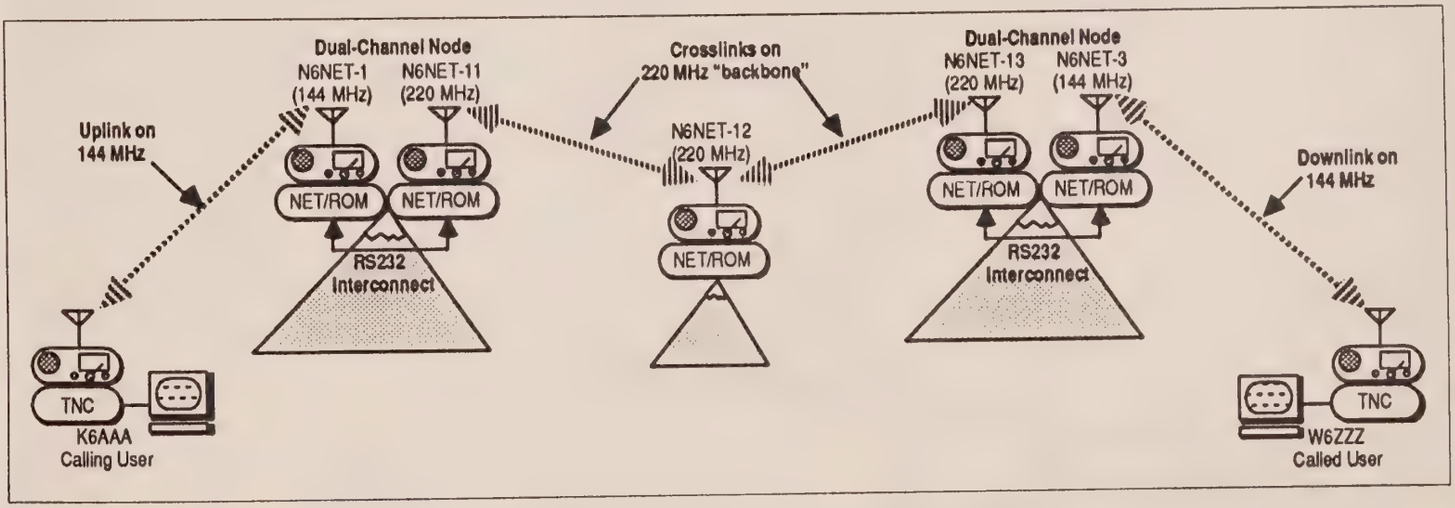
Theory of Operation

Making Connections (continued)

adjacent nodes. In all likelihood, the necessary crosslinks are already established when you issue your CONNECT command (one crosslink can carry many circuits); if not, then the necessary crosslinks are set up. All of this crosslinking stuff happens automatically and transparently—you needn't worry about it, but it's interesting to know what's going on up there on the hilltops...

Creating a "backbone" by using dual-channel nodes

To realize the full potential of NET/ROM's high-level networking capabilities, it is an excellent idea to minimize interference between local (uplink-downlink) and long-haul (crosslink) traffic. One good way to accomplish this is to reserve one radio frequency exclusively for inter-node traffic, to provide end-user access to the nodes on one or more separate frequencies, and to discourage (ideally, to prevent) end-users from using the inter-node "backbone" frequency. This approach requires network nodes that can access two or more frequencies.



NET/ROM supports such multi-channel operation without the need for exotic multi-port digipeater hardware. A dual-channel node, for example, consists simply of a pair of standard TNC-2s (with NET/ROM in each) connected together with a simple RS232 cable. Each TNC takes responsibility for handling traffic on a single frequency; cross-frequency traffic is passed over the cable between TNCs at relatively high speed. (See installation instructions for wiring of the inter-TNC cable.)

Making Connections (continued)

Configuring a multi-channel NET/ROM node for three or more channels is almost as easy as dual-channel operation. A TNC-2 with NET/ROM installed is used for each frequency. Once again, the TNCs are interconnected via their RS232 ports. Interconnecting three or more TNCs requires nothing more elaborate than some isolation diodes. (See installation instructions for details.)

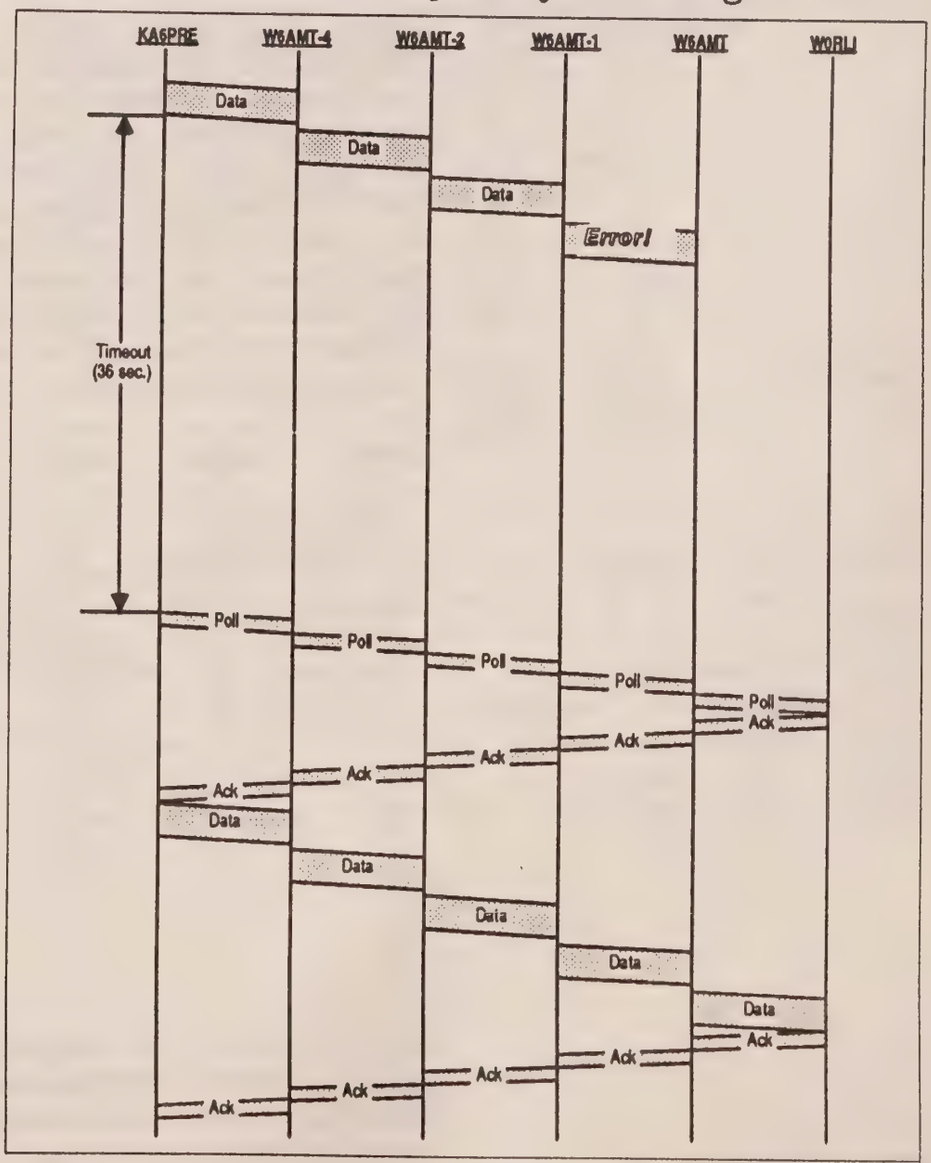
Theory of Operation

Digipeating vs. Store-and-Forward

The AX.25 protocol was originally designed for point-to-point (not digipeated) connections. AX.25 was subsequently extended to accomodate one digipeater, and later extended again to allow up to eight digipeaters.

Multi-hop digipeating

As all experienced packet radio users know, however, AX.25 is practically unusable for communications on paths exceeding two or three "hops". The reason for this is that AX.25 digipeaters do not participate in error control. For an AX.25 packet to traverse a multi-hop path, it must not fall victim to a collision or other error during any of the hops; otherwise, it must be retransmitted by the originating station and start its journey all over again.

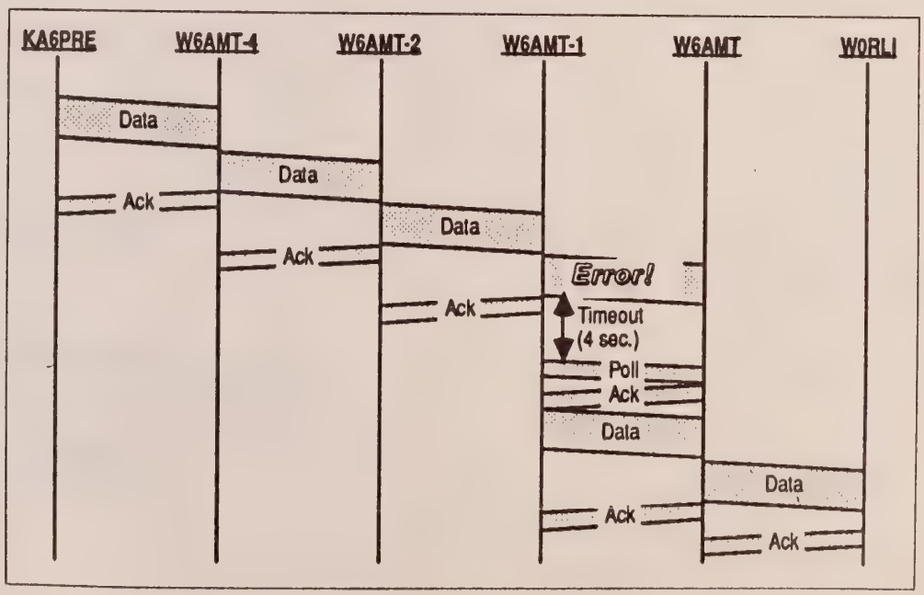


Digipeating vs. Store-and-Forward (continued)

The probability that an AX.25 packet can complete its journey successfully deteriorates rapidly as the number of hops increases. For example, it takes five hops to get from San Diego to San Francisco. If the average reliability per hop is 90% (perhaps optimistic), then the probability that a packet will traverse the five-hop path unscathed and be acknowledged (which requires ten error-free hops) is less than 35%. In other words, it will nearly three tries (on average) to get the packet through successfully. Since the usual timeout for a five-hop path is 36 seconds, the average elapsed time to get the packet through will average about 77 seconds!

Store-and-forward

Using NET/ROM nodes instead of ordinary AX.25 digipeaters changes this situation dramatically for the better. When the San Diego user transmits a packet destined for San Francisco, it is received by the local NET/ROM node serving San Diego. That node immediately passes the packet to its neighboring node to the north, and sends an acknowledgement back to the user. This process is repeated five times in all. Whenever a packet is lost due to collision or other error, recovery is handled by just the two adjacent nodes involved.

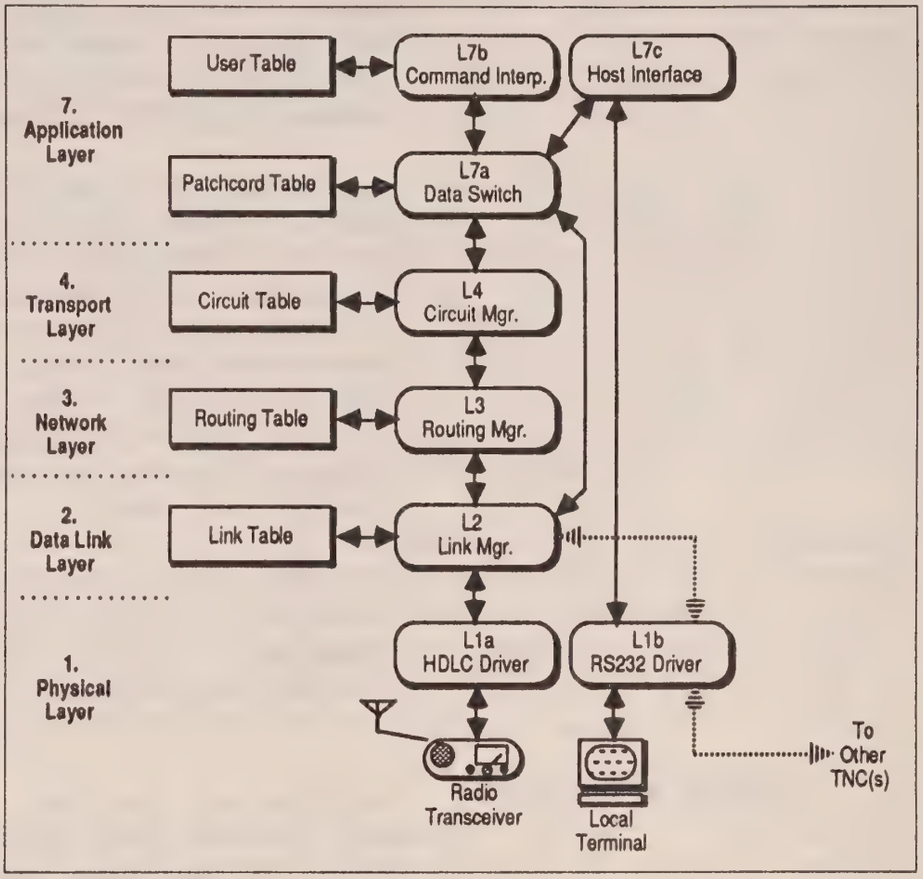


As a result, the average elapsed time to get a packet through decreases to less than 10 seconds, about an 800% improvement. For longer paths, the payoff is even more dramatic.

Theory of Operation

Firmware Organization

As shown in the accompanying diagram, the NET/ROM firmware is composed of eight major modules organized closely along the lines of the seven-layer ISO reference model.



Serial drivers (L1a & L1b)

The RS232 driver determines whether the RS232 port is attached to a terminal or to one or more other TNCs by looking at DCD-B (wired to RS232 pin 23, as described in the installation instructions). If a terminal is attached, the RS232 driver interfaces with the Host Interface module (solid arrows); if another TNC is attached, the RS232 driver interfaces with the Link Manager (dotted arrows).

Firmware Organization (continued)

Link manager (L2)

The Link Manager manages all AX.25 links (uplinks, downlinks, and crosslinks), and maintains a link table to keep track of currently-active links. This module includes most of the capabilities of a normal AX.25 TNC, including digipeating.

The AX.25 protocol provides for a "Protocol Identification" (PID) field in the header of information frames. NET/ROM identifies information that it sends over node-to-node crosslinks with a special PID value ('CF'), while information sent to or from users over uplinks and downlinks use the standard PID value ('F0'). The Link Manager passes incoming inter-node 'CF'-frames to the Routing Manager, whereas it passes incoming user 'F0'-frames directly to the Data Switch.

The Link Manager monitors traffic on each active AX.25 link. If no information has been passed over a particular link for 15 minutes, that link is automatically disconnected.

Routing manager (L3)

The Routing Manager maintains the routing table, and handles the automatic routing of crosslink traffic. It examines the network header of arriving frames to determine the callsign of the destination node. If a frame is destined for *this* node, it is passed to the Circuit Manager. Otherwise, the routing table is searched for the destination node, the next node along the route to that destination is determined, a crosslink to that node is established if none already exists, and then the frame is passed back to the Link Manager for crosslinking.

The Routing Manager also facilitates manual and automatic updates to the routing table. It supports the manual update capabilities of the NODES command, which can be used locally or remotely by an authorized control operator. It also issues automatic hourly routing broadcasts, and analyzes broadcasts received from other nodes to see if automatic routing table updates are needed.

Theory of Operation

Firmware Organization (continued)

Circuit manager (L4)

The Circuit Manager handles the initiation, disconnection, and end-to-end error and flow control for transport-layer circuits between nodes, and maintains a circuit table to keep track of currently-active circuits.

The Circuit Manager receives incoming frames from the Routing Manager. It analyzes the transport header of each frame, using a "sliding window protocol" to detect missing, duplicate, and out-of-sequence frames. It enforces circuit-layer flow control to protect the network against disproportionate loading by any one circuit. After validation, the incoming frames are passed to the Data Switch. For outgoing frames, the flow is just the reverse.

New circuits are initiated only when explicitly requested by a user (via the CONNECT command). The Circuit Manager monitors traffic on each active circuit. If no information has been passed over a particular circuit for 15 minutes, that circuit is automatically disconnected.

Data switch (L7a)

The Data Switch module acts as a switchboard within the node. It maintains a table of "patchcords" which establish two-way linkages between links, circuits, the Command Interpreter, and the Host Interface.

When a user first connects to a node (via an uplink, or via a circuit from another node), the Data Switch initially connects his uplink (or circuit) to the Command Interpreter. If the user then issues a CONNECT command to request a downlink, circuit, or host connection, then the Data Switch "patches him through" as appropriate.

Firmware Organization (continued)

Command interpreter (L7b)

The Command Interpreter analyzes and executes user commands, and diagnoses any resulting errors that may occur. When a user first enters a command, the Command Interpreter establishes a user task and makes a corresponding entry into its user table. The user remains in command mode (and his task remains active) until he disconnects or executes a successful CONNECT command. The Command Interpreter supports seven commands: CONNECT, NODES, USERS, PARMS, IDENT, RESET, and SYSOP.

The CONNECT command enables a user to request a circuit to a distant node, a downlink to a user station, or a direct connection to the local host terminal (if there is one) attached to the node.

The NODES command permits a user to obtain a list of other nodes for which the local node has routing information, and to examine the routing details to any particular node. It also allows an authorized control operator to add, delete, or change routing table entries manually.

The USERS command displays a summary of the current activity at the node, including information about active uplinks, downlinks, circuits, patchcords, and users in command mode.

The PARMS command displays a series of numeric node parameters, and allows an authorized control operator to change the values of these parameters. Likewise, the IDENT command displays the mnemonic identifier of the node, and allows a control operator to change it. The RESET command permits a control operator to reset (warm-start) the node remotely.

Theory of Operation

Firmware Organization (continued)

Command interpreter (L7b) (continued)

The SYSOP command allows an authorized control operator to establish his credentials prior to making privileged changes using the NODES, PARMS, or IDENT commands, or doing a RESET. SYSOP uses a randomized validation algorithm which makes it extremely difficult for an unauthorized user to masquerade as a control operator.

Host interface (L7c)

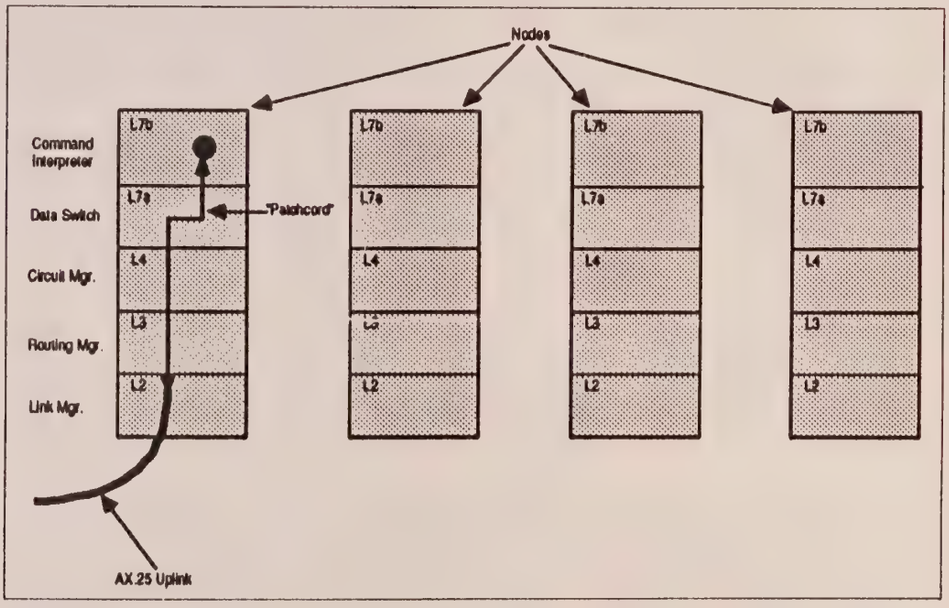
The Host Interface supports a local terminal attached to the node. It permits the local terminal to access all of the same capabilities that a remote user can access. In addition, it supports some specialized "host commands" available only from the local terminal. For example, it allows local entry of the password string used by the SYSOP command to validate control operator credentials.

Theory of Operation

Flow of Information

Let's take a more detailed look at what actually happens inside the node when you access the network. *(Warning: this section is intended mainly for the incurable hackers in the audience. If you find the going getting a bit difficult, just skip ahead to the next section. And don't worry about it...you don't really need to know this stuff!)*

First, you connect to your local node. The Link Manager acknowledges your connect request, and makes a new entry in its link table. Since it is an uplink, incoming information frames from you are passed directly to the Data Switch. Initially, the Data Switch always patches you through to the Command Interpreter. The Command Interpreter makes a new entry in its user table, and starts interpreting your commands.



Step 1—user connects to local node

Next, you send a CONNECT command that specifies the callsign or identifier of a distant node. Your command is passed from the Link Manager to the Data Switch to the Command Interpreter. The Command Interpreter asks the Circuit Manager to establish a new circuit to the distant node. If this succeeds, then the Command Interpreter asks the Data Switch to set up a "patchcord" between your uplink and the newly-created circuit. (If the circuit cannot be established for some reason, then the Command Interpreter will tell you so.)

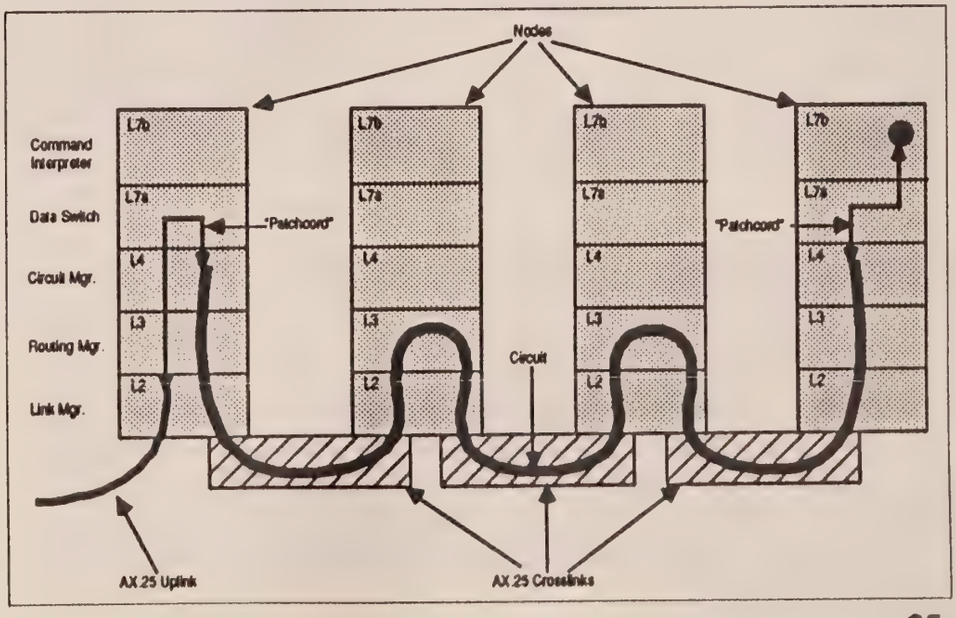
Theory of Operation

Flow of Information (continued)

To set up your circuit, the Circuit Manager creates a transport-layer connect request message, and sends it to the Routing Manager. The Routing Manager uses its routing table to determine the next node en-route to the ultimate destination. There probably is a crosslink to that node already; if not, the Routing Manager asks the Link Manager to establish one. Then, the Routing Manager passes the connect request message to the Link Manager for crosslinking.

The frame is received by the Link Manager of the next node. Since it came from a crosslink, it is passed to the Routing Manager. The Routing Manager sees that the ultimate destination hasn't been reached yet, consults its routing table to find the next en-route node, sets up a crosslink if one doesn't already exist, and passes the message on. This repeats until the destination is reached.

At the destination, the message passes from the Link Manager to the Routing Manager to the Circuit Manager. The Circuit Manager creates a transport-layer connect acknowledge message, and passes it back to the originating node via the same snake-like path (in reverse). Thereafter, any information received over the new circuit by the Circuit Manager at the distant node is passed to the Data Switch. Initially, the Data Switch patches it through to the Command Interpreter.



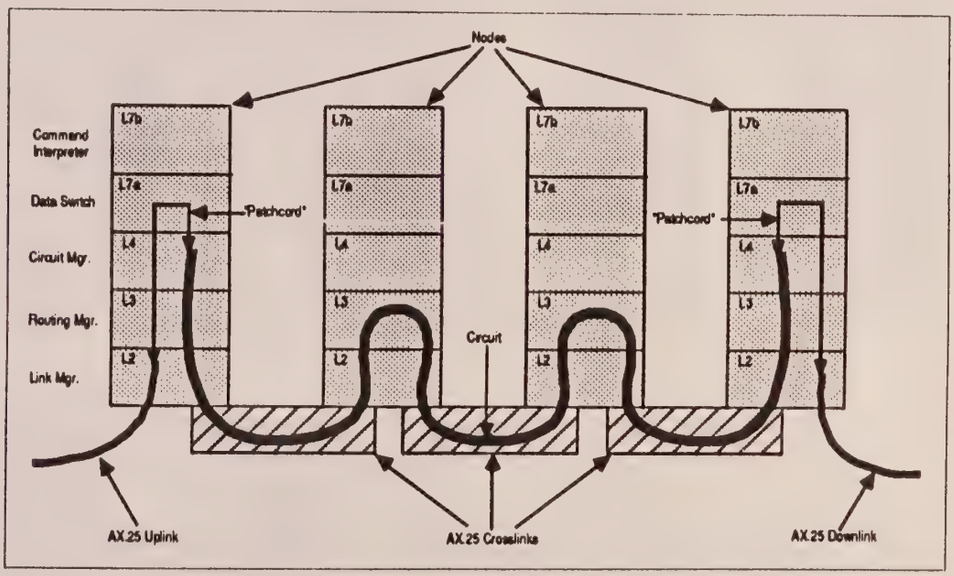
Step 2—user connects from local node to distant node

Flow of Information (continued)

Now, you send a CONNECT command that specifies the callsign of another user. Your command passes from the uplink, through the patchcord in the local node to the circuit, through the circuit (via that snake-like path through the intermediate nodes and crosslinks) to the distant node, and through the Data Switch to the Command Interpreter of the distant node.

This time, the distant Command Interpreter analyzes your CONNECT command, and asks the Link Manager to establish a new downlink to the specified user station. If this succeeds, then the Command Interpreter asks the Data Switch to set up a patchcord between your circuit and the newly-created downlink. From that point on, anything you send passes all the way to the other user station...the high-tech way! (Once again, if the downlink cannot be established for some reason, then the distant Command Interpreter will tell you so.)

Step 3—user connects from distant node to distant user



Theory of Operation

Error and Flow Control

NET/ROM uses the standard AX.25v2 link-layer packet radio protocol for crosslinks between neighboring nodes, as well as for links from each node to its local users. Normal AX.25v2 error handling is used to assure error-free transmission, and normal AX.25v2 flow control is used to control network congestion.

In addition, NET/ROM incorporates a transport-layer "sliding window protocol" to provide end-to-end error and flow control for each virtual circuit. End-to-end error control is necessary to protect against lost, duplicate, or out-of-sequence frames resulting from node failures and dynamic routing changes. End-to-end flow control is necessary to protect the network against disproportionate loading by any one circuit.

The transport-layer protocol used by NET/ROM is similar in concept to the familiar AX.25, but is somewhat more sophisticated. For example, it can accept out-of-sequence frames and re-sequence them internally. It can also selectively request a repeat of a missing frame without requiring retransmission of all succeeding frames.

Deferred Disconnect

A classic problem in packet radio occurs at the end of a QSO when one station wants to disconnect, but not before making certain that the other station has successfully received the first station's concluding information frames. This often resolves itself with a last-minute flurry of closing remarks such as "SK," "EOM," "U DISC" and so forth. NET/ROM solves this dilemma in a more elegant fashion.

If two stations are connected to one another via the network and one of the stations disconnects, NET/ROM automatically maintains its connection to the other station until all in-transit information frames have been successfully delivered to that station. NET/ROM disconnects only after all in-transit information has been delivered, or after 15 minutes has elapsed without any "forward progress" in delivering such information.

Theory of Operation

Automatic Adaptive Routing

When you ask one node for a circuit to a distant node, your CONNECT command specifies the callsign or mnemonic identifier of the destination node, but the routing is handled automatically by NET/ROM. Automatic routing is handled by the Routing Manager, and is controlled by its routing table.

The routing table within a node contains a list of all other nodes "known" to the node, together with their mnemonic identifiers. You can ask to see this list by using a parameterless NODES command. The routing table can keep track of hundreds of other nodes (limited only by the size of available memory and any constraints imposed by the control operator).

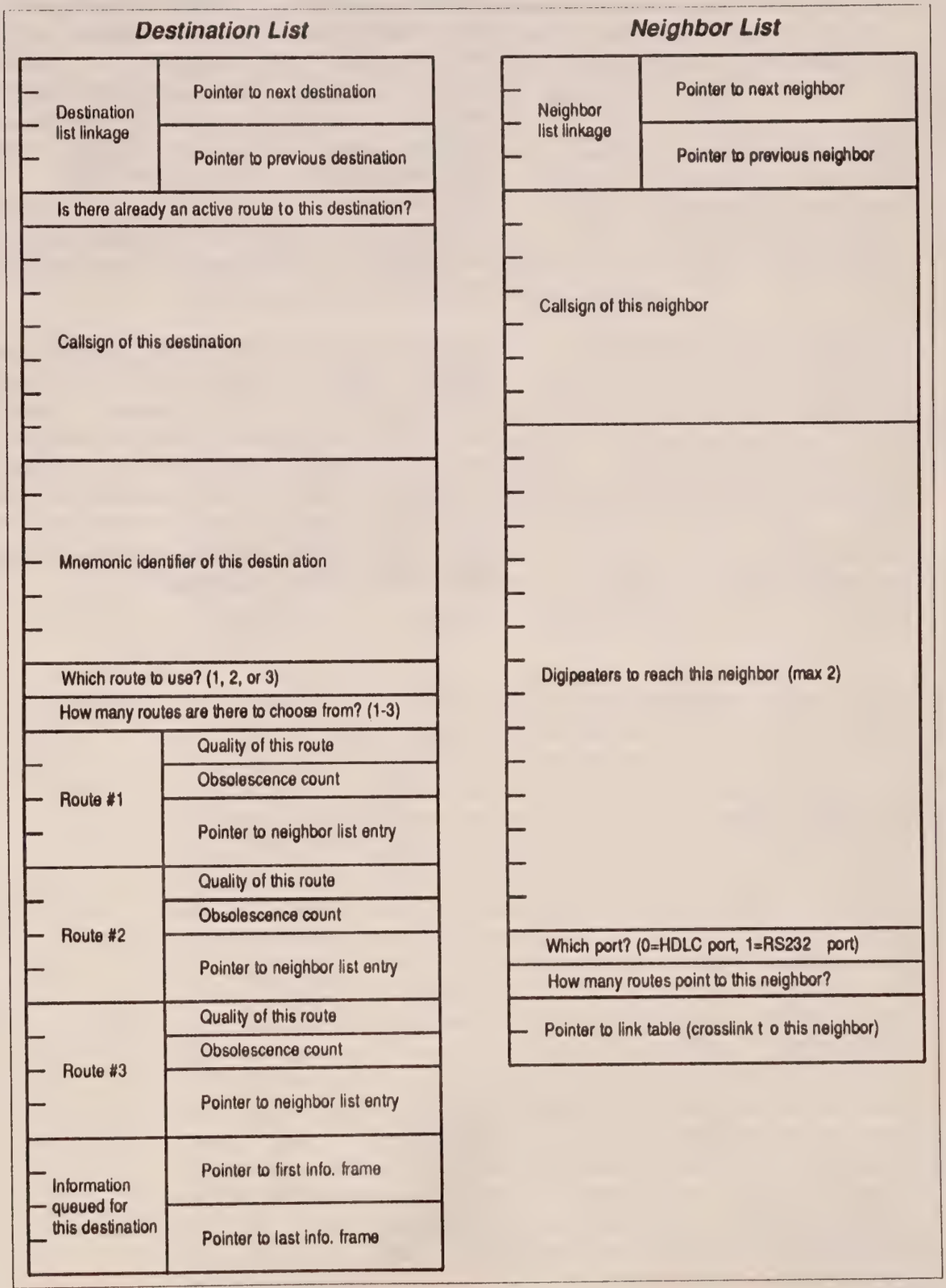
If you want to connect to an especially distant node, it is possible that your local node doesn't know of it (i.e., it is not listed in the local NODES display). In this case, you can use CONNECT to obtain a circuit to a known node nearer the desired destination, and then issue another NODES command to get a list of the nodes known to *that* node. This process can be repeated if necessary.

For each known node, the routing table can contain up to three alternate ways to route traffic to that node. The node knows the *quality* of each alternate route, and always attempts to use the "best" (i.e., highest quality) route to a destination. However, if a node or path becomes unusable due to equipment failure or poor propagation, the Routing Manager automatically switches to an alternate route (if available) to circumvent the outage. Such routing changes are handled dynamically, usually without disrupting circuits in progress.

The routing table maintained by each node consists of two dynamically allocated threaded lists: the *destination list* and the *neighbor list*. The destination list contains an entry for every other node "known" to this node. (This is the the list displayed by the NODES command.) The neighbor list contains an entry for only those "neighboring" nodes to which this node has a *direct* link.

Theory of Operation

Automatic Adaptive Routing (continued)


Destination List
Destination list linkage
Pointer to next destination
Pointer to previous destination
Is there already an active route to this destination?
Callsign of this destination
Mnemonic identifier of this destin ation
Which route to use? (1, 2, or 3)
How many routes are there to choose from? (1-3)
Route #1
Quality of this route
Obsolescence count
Pointer to neighbor list entry
Route #2
Quality of this route
Obsolescence count
Pointer to neighbor list entry
Route #3
Quality of this route
Obsolescence count
Pointer to neighbor list entry
Information queued for this destination
Pointer to first info. frame
Pointer to last info. frame
Neighbor List
Neighbor list linkage
Pointer to next neighbor
Pointer to previous neighbor
Callsign of this neighbor
Digipeaters to reach this neighbor (max 2)
Which port? (0=HDLC port, 1=RS232 port)
How many routes point to this neighbor?
Pointer to link table (crosslink t o this neighbor)

Automatic Adaptive Routing (continued)

For each node in the destination list, the routing table up to three *routes* to that destination node. In this context, a route simply identifies a neighboring node that is a step closer to the ultimate destination. For each route, the destination list maintains a *quality* value in the range 255 (best) to 0 (worst). Routes are maintained in sorted order by quality, and NET/ROM always attempts to use the highest-quality route available. It also keeps an *obsolescence count*, which enables NET/ROM to purge paths from its routing table when they have become unuseable and remained so for a protracted period of time.

Observe that the routing table does *not* contain the *entire path* to each known destination (this is unnecessary, and would require too much memory), but just a list of neighboring nodes that are reasonable choices for a next step enroute to the destination. You can ask to see the routes to a particular destination node by using a NODES command that specifies the callsign or mnemonic identifier of the destination.

Theory of Operation

Routing Table Updates

Since each node keeps track of many other nodes and the available routes to those nodes, it is important that this routing information be kept up-to-date to reflect the current state of the network. NET/ROM supports three methods of updating its routing table: *local*, *remote*, and *automatic*.

Local

When a node is first placed on-line, an initial set of routing information is entered manually by the control operator using a local terminal connected to the RS232 host port. The NODES command permits the control operator to add, delete, and modify any entry in the routing table.

Remote

The same manual update capabilities are available to the control operator at any time via remote control. To make routing table changes remotely, the control operator simply connects to the node, validates his control operator authority by means of the SYSOP command, and then updates the routing information using the NODES command. The SYSOP command uses a randomized verification algorithm that makes it extremely difficult for an unauthorized user to masquerade as a control operator.

Automatic

To enable the network to incorporate new nodes and to bypass extended outages without the need for control operator intervention, NET/ROM also provides a mechanism for automatic update of routing information on a distributed basis. Once an hour, each node automatically broadcasts a list of all other nodes which it knows how to reach—for each such node, the broadcast includes the callsign, mnemonic identifier, and the optimum route and its quality. This hourly broadcast takes the form of an AX.25 UI-frame with PID='CF'. When the broadcast is received by neighboring nodes, they automatically make any necessary additions, deletions, and modifications to their routing tables, and incorporate the revised information into their own hourly broadcasts. Thus, whenever a new node is placed on-line, knowledge of the new node automatically propagates throughout the network within a few hours. If a node or route becomes unuseable, that information also propagates automatically.

Theory of Operation

Route Quality Analysis

For each route in its routing table, NET/ROM maintains a route *quality* value in the range 255 (best) to 0 (worst). This allows it to keep alternate routes in order of quality, and to select the best available route to a destination.

A route quality value can best be visualized as a fraction (the value divided by 256) which quantifies the speed and reliability of a particular route in comparison to a theoretically perfect route (an infinitely fast and perfectly error-free path) of quality 256. For example, a route of quality 230 can be thought of as being "90% perfect" in speed and reliability (230/256 = 0.90).

The quality of each channel used by a NET/ROM node is established by the control operator of the node. As a starting point, we suggest the following values to be used as channel quality parameters:

Channel quality values

Channel Description	Quality	% Perfect
9600-baud RS232 wire interconnect (2-port)	255	99%+
9600-baud RS232 satellite interconnect (2-port)	252	98%
9600-baud RS232 wire interconnect (3-port)	248	97%
9600-baud HDLC isolated internode backbone	240	94%
1200-baud HDLC isolated internode backbone	224	88%
1200-baud HDLC user-accessed channel	192	75%
300-baud HDLC HF channel	128	50%
Known loopback or route of unknown quality	0	?

A quality value of zero is reserved by NET/ROM to identify a route of unqnown quality or a known loopback route.

The quality of a multi-segment route is simply the product of the qualities of each individual segment, where quality values are treated as fractions with an implicit denominator of 256. For example, a four-segment route that consists of two 9600-baud RS232 interconnect segments (quality 255) and two 1200-baud HDLC backbone segments (quality 224) has a calculated quality of 194:

255/256 x 255/256 x 224/256 x 224/256 = 194/256

Quality calculations are rounded to the nearest 256th.

Theory of Operation

Station Identification

In order to conform with FCC regulations concerning station identification, each NET/ROM node identifies itself every 9 minutes and 59 seconds. The station identification is carried by an AX.25 UI-frame addressed to "ID" and containing the text "Network node" plus the node's mnemonic identifier (if it has one) in parentheses. The control operator can disable this feature, if desired, by means of the PARMS command.

Digipeating

Of course, each node also supports the functions of an ordinary AX.25 digipeater. Users need not make use of the high-level networking functions of NET/ROM unless they want to. Digipeater owners can upgrade their sites to NET/ROM nodes without notifying the user population—the users won't notice any change unless they happen to monitor a station-id broadcast or try to connect to the node (expecting a "busy" message).

Furthermore, each multi-channel node is also a multi-port digipeater. Each channel is assigned a different callsign. Often, the same basic callsign will be used, but with different SSID suffixes for each frequency. (For example, N6NET-1 on 145 MHz and N6NET-11 on 220 MHz.) Cross-frequency digipeating is requested simply by including both callsigns as digipeaters (e.g., "C W6ZZZ via N6NET-1,N6NET-11,..."). If a node is assigned a mnemonic identifier, that identifier is also recognized as an AX.25 "alias". It can be used in lieu of the callsign for purposes of digipeating ("C W0RLI via LAX,SBA,MRY,SFO") or uplinking ("C LAX").

Users of distant mailboxes, gateways, and other automated servers have two options: they can connect "the old way" (via multi-hop AX.25) or "the new way" (via the uplink-crosslink-downlink procedure). Both methods will work with all existing BBSs. Auto-forwarding mailbox systems can use "the old way" until their auto-forwarding algorithms have been updated to take advantage of NET/ROM. Note that W0RLI's portable "C" bulletin board system is already capable of auto-forwarding using NET/ROM transport-layer connections.

Control Operator Validation

Authorized control operators can make manual updates to routing table entries with the NODES command, modify various node parameters with the IDENT and PARMS commands, and warm-start the node with the RESET command. To do these things remotely, a control operator must first validate his credentials by means of the SYSOP command—otherwise, the RESET command and the update functions of the NODES, IDENT, and PARMS commands are locked out.

Because packet communications between the control operator and the node can be easily monitored (and imitated) by any user, an ordinary password scheme cannot provide effective validation. Consequently, the SYSOP command utilizes a randomized validation algorithm that makes it quite difficult for an unauthorized user to masquerade as a control operator.

Here is how it works. A *password string* up to 80 characters long is entered into the node by an on-site operator via a local terminal. (The password string cannot be changed remotely.) When a remote operator enters the SYSOP command, the node replies with a list of five random numbers (each in the range 1-80). The operator must then enter the five characters that correspond to those numbered character positions in the password string. The correct characters must be entered before control operator privileges are granted.

A user who monitors this procedure can learn only five characters of the password string. He would have to monitor a large number of such procedures before he could learn enough of the password string to have any reasonable chance of a successful masquerade. The password string may contain non-printing control characters, if desired. The command interpreter does not report whether or not a SYSOP validation procedure has succeeded or failed—consequently, an especially cautious control operator could perform two or three SYSOPs, giving false answers to all but one of them, and thereby giving misleading information to anyone who might be monitoring. It also doesn't hurt to change the password string from time to time...

Commands

NET/ROM supports seven commands: CONNECT, IDENT, NODES, PARMS, RESET, SYSOP, and USERS. For any of them, the entire command verb ("CONNECT") or just a fragment ("CONN" or "CO" or "C") is allowed. Any command parameters must be separated from the verb and each other by one or more spaces. The maximum command length is 80 characters. Commands must end with a carriage-return.

CONNECT Command

The CONNECT command is used to request a circuit to another node, a downlink to another user, or a connection to the node's host terminal.

To request a circuit to another node, use:

```
CONNECT node
```

where *node* must be the callsign or mnemonic identifier of another node that is "known" to this node. (Use the NODES command to obtain a list of all known node callsigns and identifiers.) For example:

```
CONNECT WB7BNI-1
SAN:W6AMT-4} Failure with PHX:WB7BNI-1

CONNECT SFO
SAN:W6AMT-4} Connected to SFO:W6AMT
```

To request a downlink to another user, use:

```
CONNECT usercall [[VIA] digicall,…,digicall]
```

where *usercall* is the callsign of the user station, and *digicall* is the callsign (or alias) of a digipeater. If digipeaters are used, the use of "VIA" is optional ("VI" or "V" are also acceptable). Digipeaters may be separated by either spaces or commas. For example:

```
CONNECT WM6P via N6GPP-1
LAX:W6AMT-3} Busy from WM6P

CONNECT W0RLI
SFO:W6AMT} Connected to W0RLI
```

CONNECT Command (continued)

To request a connection to the node's host terminal, the command is:

```
CONNECT
```

with no parameters. For example:

```
CONNECT
SBA:W6AMT-2} Connected to SBA:W6AMT-2
```

In all cases, a successful connection is announced by the message "Connected to *callsign*". "Failure with *callsign* " indicates that the specified node or user did not respond after a number of attempts. "Busy from *callsign*" indicates that the node or user responded but refused the connection request.

Other possible error messages are "Node busy", "Circuit table full", "Link table full", and "Host table full". These messages indicate a lack of resources in the node—the user should disconnect and try again later.

An in-process CONNECT command is immediately aborted if another command or a blank line is entered before the requested connection is established.

Commands

NODES Command

The NODES command is used to display or modify the node's routing table.

To display a list of other known nodes with their mnemonic identifiers, use NODES without any parameters:

```
NODES
SAN:W6AMT-4} Nodes:
LAX:W6AMT-3      SBA:W6AMT-2      PRB:W6AMT-1
SFO:W6AMT        RBL:W6AMT-7      ONT:AA6TN-1
LAS:K7WS-1       PHX:WB7BNI-1
```

The normal NODES display includes all known nodes in the routing table *except* "hidden nodes" that have mnemonic identifiers that start with the character "#". To obtain a display of all nodes *including* such hidden nodes, use the command "NODES *":

```
NODES *
SAN:W6AMT-4} Nodes:
LAX:W6AMT-3      SBA:W6AMT-2      PRB:W6AMT-1
#PRB:W6AMT-11    SFO:W6AMT        #SFO:W6AMT-10
#WWORM:WB6FFC-1 #EWORM:WA3YMH-1 RBL:W6AMT-7
ONT:AA6TN-1      LAS:K7WS-1       PHX:WB7BNI-1
```

To display specific routing information for a particular node, use NODES followed by the callsign or identifier of the node in question:

```
NODES PHX
LAX:W6AMT-3} Routes to PHX:WB7BNI-1
> 108  6  0 AA6TN-1
  81  6  0 W6AMT-4
```

This command shows up to three routes to the specified node. For each route, the following items are displayed:

- the symbol ">" if the route is in use
- quality of the route (255 is best, 0 is worst)
- obsolescence count (0 denotes a locked entry)
- port number (0=HDLC port, 1=RS232 port)
- path to neighboring node (callsign + any digis)

Commands

NODES Command (continued)

The NODES command also supports manual updates to the routing table, but this capability is available only to a control operator who has previously validated his credentials during this connection by successfully executing the SYSOP command. To add or delete routing table entries, the commands are:

```
NODES nodecall + ident quality count port neighbor [digicall...]
NODES nodecall - ident quality count port neighbor [digicall...]
```

The "+" version adds a new entry to the routing table; the callsign *nodecall* is added to the list of known nodes if it isn't already there. The *ident* parameter is a mnemonic identifier up to six characters long, or "*" if the node has no identifier. The *quality* parameter is a decimal integer in the range 255 (best) to 0 (worst) that defines the quality of the best-case route to the destination node via the specified neighbor. The *count* parameter is the initial obsolescence count—*zero* denotes a permanent, unchangeable route entry. The *port* parameter is given as either 0 (HDLC port) or 1 (RS232 port). The *neighbor* and *digicall* parameters give a path to an adjacent node that is enroute to the destination (two digipeaters max).

The "-" version searches the routing table for an entry that exactly matches the given *nodecall*, *port*, *neighbor* and *digicall* parameters, and deletes it if such an entry is found; deleting the last route for a particular *nodecall* also removes it from the list of known nodes. In case of an invalid request (or a valid request by an unvalidated user), these commands are quietly ignored, not diagnosed. Examples:

```
NODES WB7BNI-1 + PHX 115 6 0 AA6TN-1
LAX:W6AMT-3} Routes to PHX:WB7BNI-1
> 115 6 0 AA6TN-1
  81 6 0 W6AMT-4

NODES WB7BNI-1 - PHX 81 6 0 W6AMT-4
LAX:W6AMT-3} Routes to PHX:WB7BNI-1
> 115 6 0 AA6TN-1
```

Commands

USERS Command

The USERS command displays a summary of who is using the node:

```
USERS
LAX:W6AMT-3} NET/ROM 1.0 (720)
Uplink(WA8DED)                <--> Host(LAX:W6AMT-3)
Uplink(KA6SOX)                <--> Downlink(KA6SOX-15 WB6YMH)
Uplink(NK6K-2)                <--> Circuit(SBA:W6AMT-2 NK6K-2)
Circuit(SFO:W6AMT WB6ASR)     <--> Downlink(WB6ASR-15 WD6E)
Circuit(SFO:W6AMT WB6RAL)     <--> Circuit(LAS:K7WS-1 WB6RAL)
Circuit(SBA:W6AMT-2 WA6JPR)
Uplink(NK6K)
```

The heading of the USERS display indicates the version of the NET/ROM firmware in use at the node, and the amount of free RAM space (shown in parentheses, and expressed as a number of 32-byte buffer segments).

After the heading, the USERS display shows the active circuits and links, using the following formats:

- Uplink(*fromcall*)
- Downlink(*fromcall tocall*)
- Circuit(*othernode usercall*)
- Host(*thisnode*)

The "<-->" symbols represent active "patchcords" in the node that connect uplinks, downlinks, circuits, and possibly the host terminal (if any). The lines which do not contain the patchcord symbol represent users who are in command mode at the node.

SYSOP Command

The SYSOP command allows an authorized control operator to establish his credentials prior to making privileged changes using the NODES, PARMS, or IDENT commands, or executing a RESET command. It uses a randomized validation algorithm, in conjunction with a "password string" previously entered via the node's host terminal. The command is simply "SYSOP", in response to which the node will respond with a list of five random numbers:

```
SYSOP
LAX:W6AMT-3} 26 13 54 5 38
```

The control operator must respond by entering the five characters in the correspondingly numbered character positions of the "password string". The random numbers returned by SYSOP will always correspond to valid non-space character positions of the password string. The five required characters may be entered with or without intervening spaces, and must be followed by a carriage-return. There is no acknowledgement of success or failure.

For example, assume the password string is "The quick brown fox jumped over the lazy dog's back 0123456789 times". A valid control operator validation sequence might be:

```
SYSOP
LAX:W6AMT-3} 26 13 54 5 38
dolqa
```

where the 26th character of the password string is "d", the 13th character is "o", etc. If the validation succeeds, subsequent IDENT, NODES and PARMS update requests and the RESET command are honored; otherwise, they are quietly ignored. When accessing the node from a local host terminal, control operator privileges are automatically granted. There is no need to use the SYSOP command in this case.

Commands

IDENT Command

The IDENT command allows an authorized control operator to set or change the mnemonic identifier of the node. The command is:

```
IDENT identifier
```

where *identifier* is a string up to six characters long, or "*" if the node is to have no identifier at all. For example:

```
IDENT LAX
LAX:W6AMT-3} LAX
IDENT *
W6AMT-3}
```

Node identifiers should normally be composed of letters and digits only. Non-printing characters and punctuation marks are invalid (with one exception discussed in the next paragraph). Lower-case letters are converted to upper-case. In addition, NET/ROM will not accept a node identifier that "looks like" a valid amateur callsign—a string of letters and digits looks like a callsign if it is four to six characters long, has either one or two digits, and the rightmost digit is neither the first nor the last character in the string.

A node identifier may use the character "#" in its first character position. This causes the node to be supressed from the NODES displays at other nodes. (*NOTE:* The use of other punctuation characters in node identifiers is reserved for possible future extensions of NET/ROM.)

Before using the IDENT command to change the node identifier remotely, you must validate your credentials as a control operator by using the SYSOP command...otherwise, the node identifier is left unchanged.

PARMS Command

The PARMS command can be used to display or change various numeric parameters that affect operation of the node. To display the node parameters, use PARMS with no parameters:

```
PARMS
LAX:W6AMT-3} 50 1 192 255 6 5 3600 64 60
3 3 180 4 4 900 16 4 7 10 100 18000 1 1 1
```

The following 24 node parameters are displayed in sequence:

No.	Description of Parameter	Default	Min	Max
1.	Max destination list entries	50	1	400
2.	Worst quality for auto-updates (0 disables)	1	0	255
3.	Channel 0 (HDLC) quality	192	0	255
4.	Channel 1 (RS232) quality	255	0	255
5.	Obsolescence count initializer	6	0	255
6.	Obsolescence count min. to be broadcast	5	1	255
7.	Auto-update broadcast interval (sec)	3600	0	65535
8.	Network "time-to-live" initializer	64	0	255
9.	Transport timeout (seconds)	60	5	600
10.	Transport maximum tries	3	2	127
11.	Transport acknowledge delay (seconds)	3	1	60
12.	Transport busy delay (seconds)	180	1	1000
13.	Transport requested window size (frames)	4	1	127
14.	Congestion control threshhold (frames)	4	1	127
15.	No-activity timeout(seconds)	900	0	65535
16.	Link digipeater wait "DWAIT" (10ms inc)	16	0	127
17.	Link T1 timeout "FRACK" (seconds)	4	1	15
18.	Link tx window size "MAXFRAME" (frames)	7	1	7
19.	Link maximum tries (0=try forever)	10	0	127
20.	Link T2 timeout (10ms increments)	100	0	65535
21.	Link T3 timeout (10ms increments)	18000	0	65535
22.	AX.25 digipeating (1=enabled, 0=disabled)	1	0	1
23.	Validate callsigns (1=enabled, 0=disabled)	1	0	1
24.	Station ID beacons (1=enabled, 0=disabled)	1	0	1

To change node parameters, use PARMS followed by a series of decimal values in the same sequence described above:

```
PARMS * * 224 * 8 6
LAX:W6AMT-3} 50 1 224 255 8 6 3600 64 60
3 3 180 4 4 900 16 4 7 10 100 18000 1 1 1
```

To change a particular parameter, you must enter values for all preceding parameters as well. Entering "*" instead of a value causes the corresponding parameter value to be left unchanged. If fewer values are given then there are parameters,

Commands

PARMS Command (continued)

then the trailing parameters are also left unchanged. Before using the PARMS command to change node parameters remotely, you must validate your credentials as a control operator by using the SYSOP command...otherwise, the node parameters are left unchanged.

RESET Command

The RESET command allows an authorized control operator to perform a warm-start reset of the NET/ROM firmware. The command is:

```
RESET 32767
```

Any other form of the RESET command is ignored. Before using this command remotely, you must validate your credentials as a control operator by using the SYSOP command...otherwise, the RESET is ignored.

WARNING: This command is dangerous, and should be used only as a last resort. The RESET command instantly terminates all links and circuits at the node (including the control-operator's own connection).

Command Interpreter Messages

The following messages may be displayed by the command interpreter:

```
Connected to callsign
Busy from callsign
Failure with callsign
Invalid command (CONNECT IDENT NODES PARMS USERS)
Invalid callsign
Node busy
Circuit table full
Link table full
Host table full
Routing table full
```

(Intentionally left blank.)

Host Interface

The node's host terminal may be used to enter both host command lines and information lines. Lines may be up to 256 characters long, including the terminating carriage-return. BS or DEL may be used to delete the last one or more characters typed on a line; CTRL-U or CTRL-X may be used to delete the entire line.

Lines that don't start with an ESC are interpreted as information lines. If the host interface is connected, information lines are sent across the connection; otherwise, they are discarded.

Host command lines begin with an ESC character (echoed as "* "). Valid host commands are: C, D, P, T, and Y.

ESC-C — connect

The ESC-C command connects the host terminal to the node. When connected, the host terminal acts like a user that has been uplinked to the node. All ordinary node commands (CONNECT, IDENT, NODES, PARMS, RESET, SYSOP, and USERS) may then be entered as information lines. However, the host connection automatically has control operator privileges, so the SYSOP command is unnecessary.

NOTE: The host terminal connection (like any uplink) is automatically disconnected after 15 minutes of no activity.

ESC-D — disconnect

The ESC-D command disconnects the host terminal from the node.

NOTE: For unattended operation, be sure to disconnect the host interface with ESC-D!

ESC-P — password string

The ESC-P command sets the "password string" used by the SYSOP command to validate the credentials of control operators. The phrase may be up to 80 characters long (any excess is ignored), and may include spaces and control characters (except for CR and LF). Upper- and lower-case letters are treated as distinct from one another in the password string.

ESC-P — password string (continued)

Note that the random numbers returned by the SYSOP command will always correspond to valid non-space character positions of the password string. For maximum security, it is a good idea to use a password string that is close to the maximum length of 80 characters, and doesn't contain too many spaces.

ESC-P with no parameter displays the current password string.

EST-T — transmitter key-up delay

The ESC-T command sets the transmitter key-up delay (TXDELAY), which is the time interval allowed by the node between asserting push-to-talk and starting to transmit data on the HDLC port. ESC-T expects a decimal integer parameter in the range 0–255 which defines the delay in 10-millisecond increments. The default value is 30 (i.e., 300 milliseconds).

ESC-T with no parameter displays the current setting.

ESC-Y — host connections enable/disable

ESC-Y1 enables the host interface for incoming connections. ESC-Y0 disables the host interface for incoming connections (default). ESC-Y displays the current host interface state (0 or 1).

NOTE: For unattended operation, be sure to disable the host interface with ESC-Y0!

Host Interface Messages

The following messages may be displayed by the host interface:

```
CONNECTED to callsign
DISCONNECTED fm callsign
CONNECT REQUEST fm callsign
INVALID COMMAND
```

Installation

This section provides a step-by-step checklist for installing a new NET/ROM node. For a hassle-free installation, be sure to follow the instructions carefully.

Hardware Setup

Verify TNC-2 rev. 2 PCB

NET/ROM requires hardware using TAPR TNC-2 revision 2 (or later) circuitry. Late-model TAPR kits and all commercially manufactured "clones" use this circuitry. However, a few early-model TAPR kits were shipped with the version 1 circuit board, and these must be upgraded with the version 2 modifications available from TAPR.

Upgrade to 32K of RAM

For most installations, NET/ROM also should be installed in a TNC-2 with 32K of RAM. It will run with 16K, but will run short of memory under moderate-to-heavy traffic loads.

Most TNC-2s manufactured prior to 1987 were originally equipped with only 16K of RAM. Inasmuch as the latest version of TAPR firmware requires 32K, however, many TNC-2s have already been upgraded to 32K.

To upgrade a 16K TNC-2 to 32K:

❶ Remove U24 and U25 (both are 8K RAMs, type 6264).

❷ On the bottom of the circuit board, cut the trace between the center and top positions of JMP12.

❸ Add a wire connecting the center and bottom positions of JMP12.

❹ Install one 32K RAM IC (type 43256 or 62256) in U25. (If not available locally, may be obtained from TAPR for $20.00.)

Hardware Setup (continued)

Increase time constant of PTT failsafe timer

All tested TNC-2s have a push-to-talk failsafe timer which limits key-down time to approximately 10–12 seconds maximum. This value is too short for NET/ROM use at 1200 baud, in our experience, and can cause truncation of crosslink transmissions and routing broadcasts. We strongly recommend increasing the failsafe timeout interval to approximately 60 seconds. To do this, simply replace capacitor C31 with a 47 microfarad radial-lead electrolytic capacitor rated at 15 WVDC or higher. Be sure to observe proper polarity.

Set CPU clock to 4.9 MHz

TNC-2's are normally set up for slow CPU clock speed (2.4576 MHz). Although NET/ROM will run reliably at this speed, performance is noticeably improved by changing to high CPU clock speed (4.9152 MHz). Although this is somewhat faster than the rated speed of some TNC-2 parts, we have tested NET/ROM with numerous TNC-2s (including clones from several manufacturers), and none failed to operate reliably at the higher clock speed.

To change a TNC-2 to high clock speed:

❶ On the bottom of the circuit board, cut the trace connecting the center and right positions of JMP2.

❷ Add a wire connecting the center and left positions of JMP2.

NOTE: For the MFJ 1270B, the jumpering is different. This may also be true for other TNC-2 "not-quite-clones" so check your TNC documentation carefully.

Installation

Hardware Setup (continued)

Wire DCD-B to RS232 port

The following modification is required for TNCs that will be used in dual- or multi-channel NET/ROM configurations. Since it does not impair normal operation in any way, we recommend it for all TNCs to be used with NET/ROM...you may want to upgrade to dual-channel operation later.

To make this modification, connect one end of a wire to pin 23 of the RS232 connector. Connect the other end of the wire to pins 1-2-3 of JMP9 (these three pins are already hooked together on the circuit board).

This modification allows the NET/ROM firmware to be configured for multi-channel operation by jumpering RS232 pins 10 and 23 together in the TNC-to-TNC cable.

Upgrade U3 op-amp

Most TNC-2s use an LM324 op-amp (U3) to generate bipolar RS232 output signals. In our experience, most LM324s do not have sufficient slew rate for reliable operation at 9600 baud. This can be remedied easily, and we strongly recommend the fix.

Simply replace the LM324 at U3 with either a TL074 or TL084 IC (manufactured by Texas Instruments and perhaps others).

Set baud-rate switches

Follow the switch-setting instructions in the TNC-2 manual. We have performed extensive reliability testing of NET/ROM at HDLC and RS232 baud rates up to 9600; it runs reliably at these high baud rates even at slow CPU clock speed, although we strongly advise using 4.9 MHz CPU clock for reasons of efficiency.

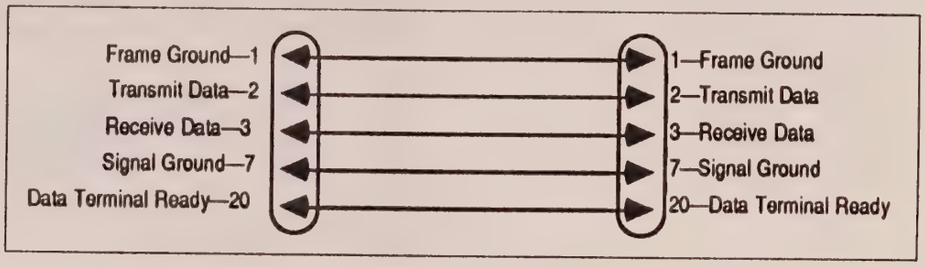
For dual- or multi-channel operation, we strongly suggest setting the RS232 speed at 9600 baud—it makes a big difference in cross-channel performance! Be sure to upgrade the U3 op-amp, however.

Installation

Hardware Setup (continued)

Connect host terminal

To connect a host terminal to the TNC-2, almost any standard RS232 cable will do (as long as pins 9, 10, and 23 are not used). If you are wiring a special cable for this purpose, wire it as follows:



It is also OK to leave DTR (pin 20) unconnected. Be certain to leave pins 9, 10, and 23 unconnected (at the TNC end, at least).

Make sure the TNC still works!

Connect a terminal to the TNC, power it up, and make sure that you get a sign-on message and that the unit still appears healthy. If it doesn't, then either (1) you have made an error; (2) you have installed a bad IC; or (3) your TNC won't run at the fast CPU clock speed.

Finally, install NET/ROM

NET/ROM is distributed in the form of a 27C256 EPROM which simply plugs into the ROM socket (U23) of the TNC-2 in place of the standard TAPR firmware ROM. Be very careful when inserting the new EPROM, making sure that pin 1 is oriented correctly, and that none of the pins are bent under the IC.

The node callsign is "hard-coded" into each NET/ROM EPROM, and cannot be changed. If you must change the node's callsign, you will have to order a new EPROM.

NOTE: Be certain to save the original ROM—you may need it if you ever want to recalibrate the TNC-2 modem or to restore regular TNC functionality.

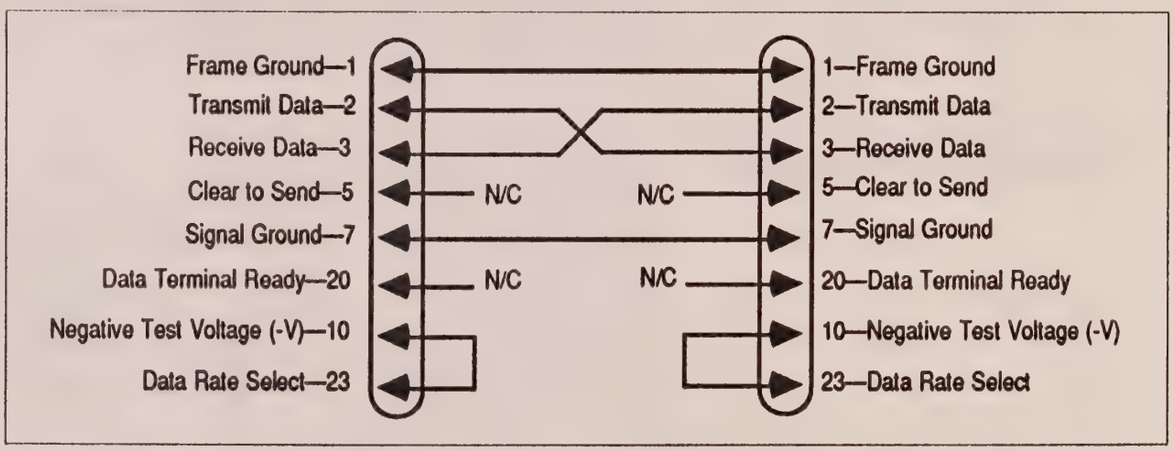
Installation

Parameter Setup	After setting up the TNC-2 hardware, connect a terminal to the RS232 port. Power up the TNC-2 and make sure you see the NET/ROM sign-on message. Then perform the following steps:
Verify the node's callsign	The callsign of the node (as encoded into the EPROM) is displayed in the NET/ROM sign-on message. Make sure it is correct!
Connect to the node	Enter ESC-C followed by a carriage-return. This connects the host terminal to the node's command interpreter. You should see the following message: "CONNECTED to *nodecall*".
Enter the node's identifier	Use the IDENT command to enter the mnemonic identifier of the node, which can be up to six characters long. Don't use punctuation or non-printing control characters in the identifier, and don't use an identifier that "looks like" a valid amateur callsign. (Suggestion: three-letter airport identifiers make nice node mnemonics.)
Enter a password string	Enter ESC-P followed by a "password string" up to 80 characters long. It is best to pick a string that occupies the full 80 characters, or close to it, and doesn't contain too many spaces. The string may contain any ASCII characters except CR and LF...even non-printing control characters are legal. You must remember the password string in order to perform privileged control operator functions remotely. Don't forget it unless you enjoy trips to the site!
Prepare the node for unattended operation	Enter ESC-D to make sure the host interface is disconnected. Enter ESC-Y0 to disable host connections. You should always remember to do these two things before you disconnect the host terminal.

Dual- or Multi-Channel Operation

To install a dual- or multi-channel NET/ROM node, you should follow the previously-described steps for each TNC. Make certain that each TNC has a different callsign (often, only the SSID suffix changes). Verify that each TNC is functioning correctly as a single-channel node before attempting to interconnect the TNCs.

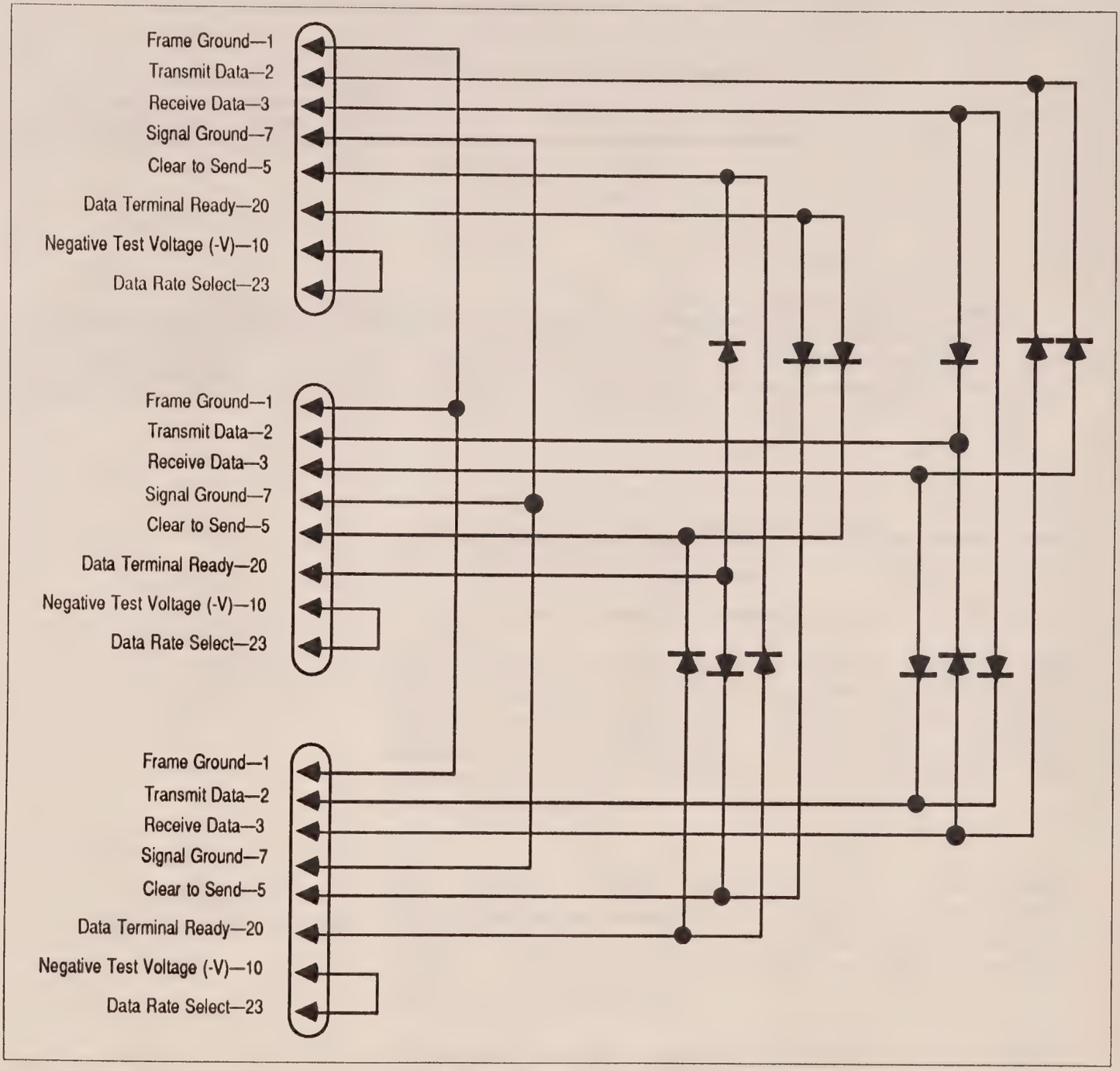
For a dual-channel node, simply connect the two TNCs together, using a special RS232 cable wired as shown in the diagram. (Don't try to use the host-terminal cable—it won't work!)



To interconnect three or more TNCs as a multi-channel node, you will have to make up a diode-matrix coupler. A schematic for a three-channel coupler is shown on the facing page. It uses 12 diodes (1N4148 or equivalent). A four-port coupler is similar, but requires 24 diodes, and is probably the maximum practical configuration. [Hint: the formula for the number of diodes is *2*N*(N-1)*.]

Installation

Dual- or Multi-Channel Operation (continued)

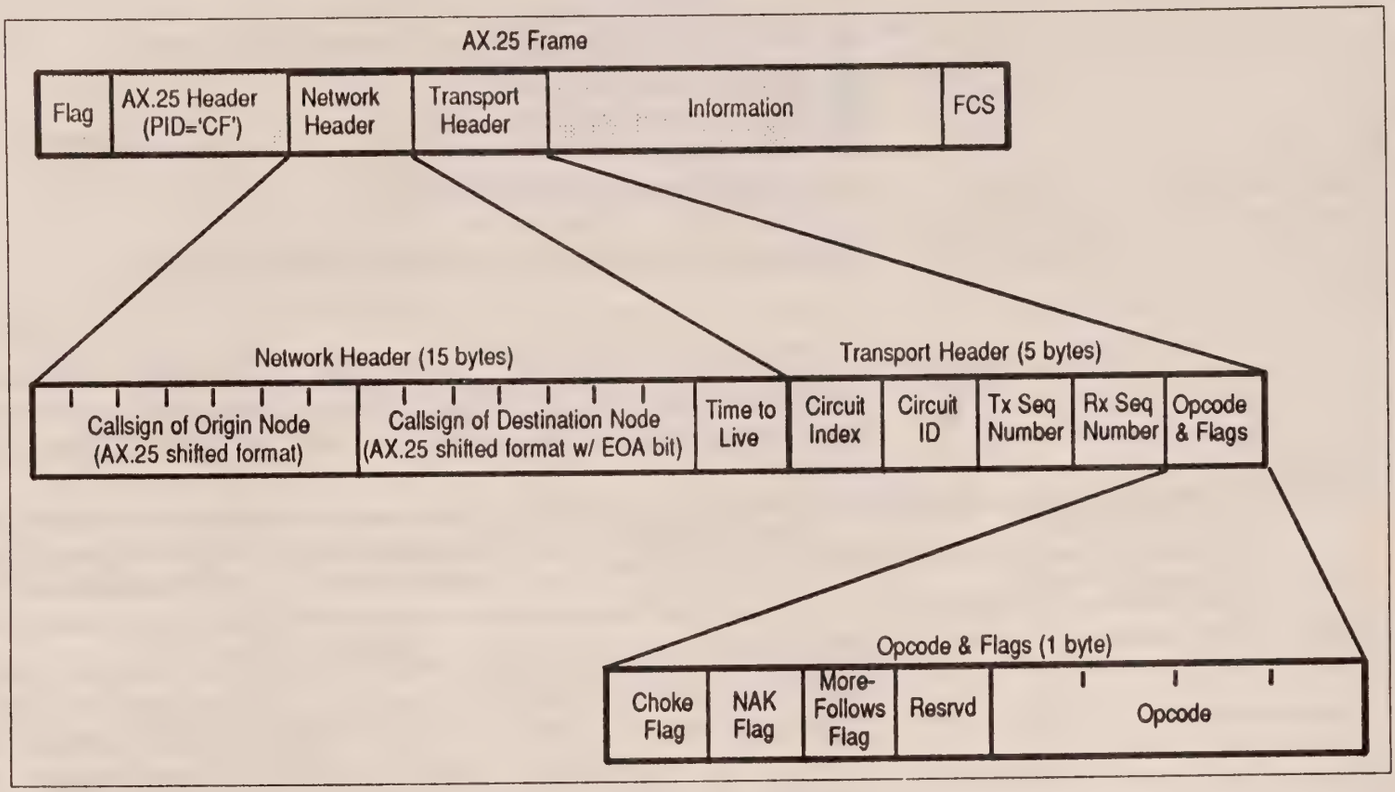


(Intentionally left blank.)

Appendix A — Protocols

Inter-Node HDLC Frame Structure

The HDLC frame structure used by NET/ROM for its inter-node crosslinks is illustrated in the following diagram:



Each frame consists of a standard AX.25 link header, followed by a 15-byte network header and a 5-byte transport header. The network header contains the information needed for automatic routing, while the transport header supports end-to-end error and flow control for circuits

Transport-Layer End-to-End Protocol

The Circuit Manager implements a conventional "sliding window protocol" in order to provide end-to-end flow and error control for each transport circuit. The protocol is similar to AX.25, but with these major differences:

❶ Receive window size is negotiated, and usually greater than 1.

❷ 8-bit sequence numbers, allowing window sizes up to 127 frames.

❸ Selective NAKing is supported.

Transport-Layer End-to-End Protocol (continued)

Six different transport-layer message types are supported:

❶ Connect Request
❷ Connect Acknowledge
❸ Disconnect Request
❹ Disconnect Acknowledge
❺ Information
❻ Information Acknowledge

These are described in greater detail below.

Connect request

A Connect Request is used to initiate a new transport circuit between the originating node and the destination node, on behalf of the originating user.

My Circuit Index	My Circuit ID	(Unused)	(Unused)	Opcode = 'x1'	Propose Window Size	Callsign of Originating User (AX.25 shifted format)	Callsign of Originating Node (AX.25 shifted format)

The *circuit index* is the subscript of a circuit table entry in the originating node. The *circuit ID* is a serial number used to qualify the circuit index, in order to eliminate any possible ambiguity in identifying the circuit. The *proposed window size* specifies the maximum receive window size (in frames) that the originating node is prepared to accomodate.

Connect acknowledge

Your Circuit Index	Your Circuit ID	My Circuit Index	My Circuit ID	Opcode = 'x2'	Accept Window Size

A Connect Acknowledge is used to respond to an incoming Connect Request. If the high-order bit of the *opcode* byte is set, it indicates that the Connect Request is being refused; otherwise, it is being accepted. The *accepted window size* indicates the negotiated size of the receive window for this circuit, and will never exceed the *proposed window size* of the Connect Request.

Appendix A — Protocols

Transport-Layer End-to-End Protocol (continued)

Disconnect request

Your Circuit Index	Your Circuit ID	(Unused)	(Unused)	Opcode = 'x3'

A Disconnect Request is used to request the termination of a transport circuit, and may be sent by the node at either end of the circuit.

Disconnect acknowledge

Your Circuit Index	Your Circuit ID	(Unused)	(Unused)	Opcode = 'x4'

A Disconnect Acknowledge is used to acknowledge a Disconnect Request.

Information

An Information message is used to pass user information across a transport circuit.

Your Circuit Index	Your Circuit ID	Tx Sequence Number	Rx Sequence Number	Opcode = 'x5'	Information

Because AX.25 frames are limited to 256 bytes and the combined network and transport header overhead totals 20 bytes, the maximum size of the information field is 236 bytes. NET/ROM automatically fragments and reassembles any user-supplied link-layer information frames that exceed 236 bytes in order to meet this constraint.

Each Information message also serves as a "piggybacked" Information Acknowledge. The *Tx Sequence Number* identifies the current information, and the *Rx Sequence Number* specifies the next incoming information expected.

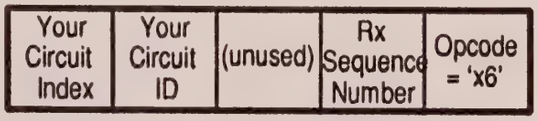
If the *choke flag* is set (bit 7 of the *opcode* byte), it indicates that this node cannot accept any more Information messages until further notice. If the *NAK flag* is set (bit 6 of the *opcode* byte), it indicates that a selective retransmission of the frame

Transport-Layer End-to-End Protocol (continued)

Information (continued)

identified by the *Rx Sequence Number* is being requested. If the *more-follows flag* is set (bit 5 of the *opcode* byte), it indicates that the information is a fragment of a long information frame, and must be reassembled with one or more following information messages by the destination node.

Information acknowledge



An Information Acknowledge is used to acknowledge incoming Information messages. The *Rx Sequence Number* specifies the next incoming information expected.

If the *choke flag* is set (bit 7 of the *opcode* byte), it indicates that this node cannot accept any more Information messages until further notice. If the *NAK flag* is set (bit 6 of the *opcode* byte), it indicates that a selective retransmission of the frame identified by the *Rx Sequence Number* is being requested.

Appendix A — Protocols

RS232 Interconnect Protocol

For multi-channel nodes, information is passed among the interconnected TNCs via their RS232 ports. The interconnect operates using the same link-, network-, and transport-layer protocols as normal HDLC crosslinks, except that a simple asynchronous variant of HDLC is employed.

Each frame is preceded with an ASCII STX (instead of an HDLC flag), and terminated by an ETX plus a one-byte checksum (instead of an HDLC frame-check sequence). Any embedded STX, ETX, or DLE characters within the body of the frame are prefixed by a DLE character (this takes the place of normal HDLC "bit stuffing").

When two TNCs are connected using the recommended TNC-to-TNC cable, communications between the two TNCs is full-duplex and extremely fast (especially at 9600 baud).

When three or more TNCs are interconnected using the recommended diode-matrix coupler circuit, intercommunications between the TNCs is essentially half-duplex and utilizes CSMA/CD arbitration. Consequently, do not expect performance to be quite as spectacular as it is with the dual-channel configuration.

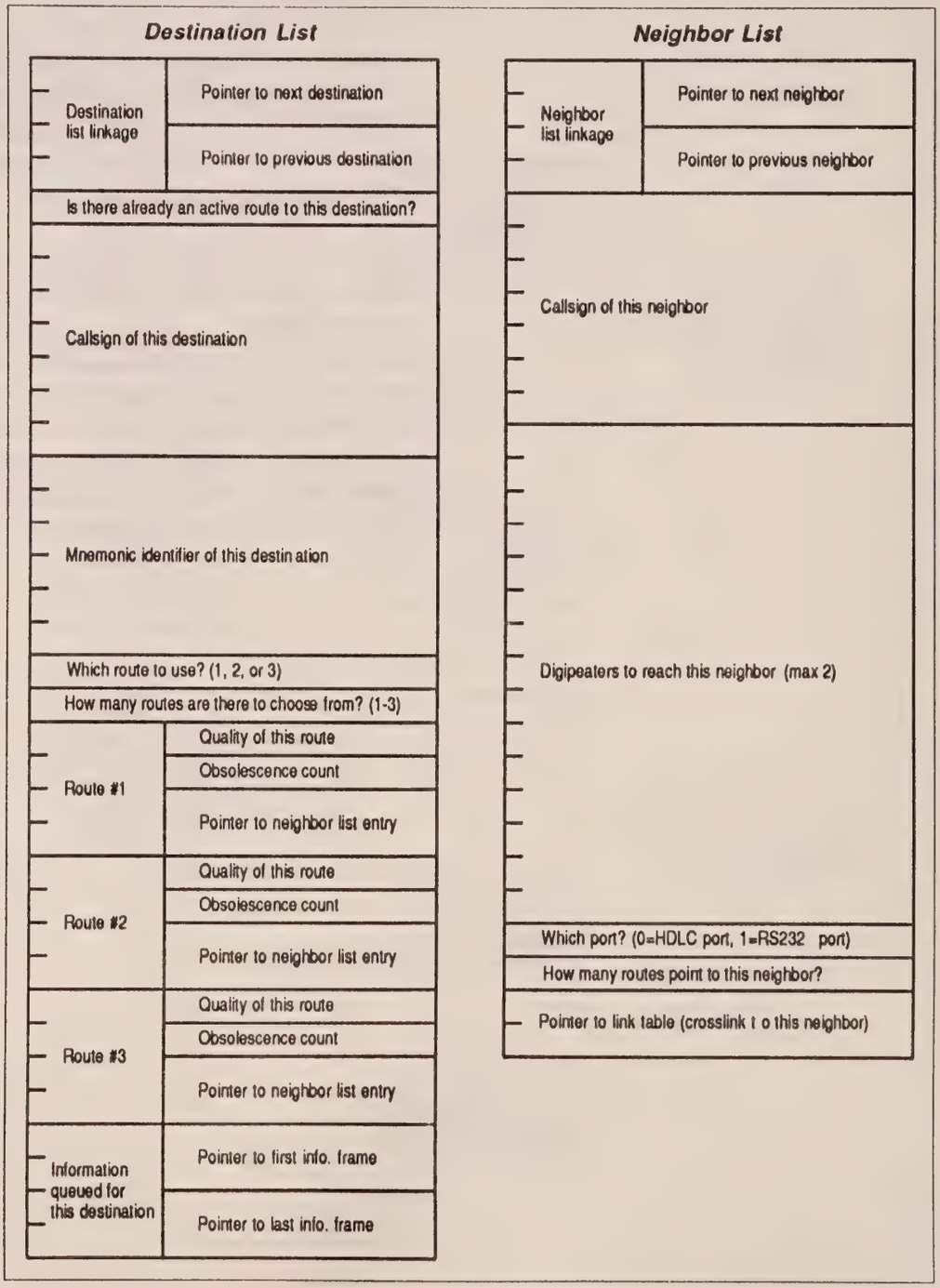
Appendix A — Protocols

(Intentionally left blank.)

Appendix B — Routing

Routing Table Structure

The routing table maintained by each node consists of two dynamically allocated threaded lists: the *destination list* and the *neighbor list*. The destination list contains an entry for every other node "known" to this node. (This is the the list displayed by the NODES command.) The neighbor list contains an entry for only those "neighboring" nodes to which this node has a *direct* link.


Destination List
Destination list linkage
Pointer to next destination
Pointer to previous destination
Is there already an active route to this destination?
Callsign of this destination
Mnemonic identifier of this destin ation
Which route to use? (1, 2, or 3)
How many routes are there to choose from? (1-3)
Route #1
Quality of this route
Obsolescence count
Pointer to neighbor list entry
Route #2
Quality of this route
Obsolescence count
Pointer to neighbor list entry
Route #3
Quality of this route
Obsolescence count
Pointer to neighbor list entry
Information queued for this destination
Pointer to first info. frame
Pointer to last info. frame
Neighbor List
Neighbor list linkage
Pointer to next neighbor
Pointer to previous neighbor
Callsign of this neighbor
Digipeaters to reach this neighbor (max 2)
Which port? (0=HDLC port, 1=RS232 port)
How many routes point to this neighbor?
Pointer to link table (crosslink t o this neighbor)

Appendix B — Routing

Routing Table Structure (continued)

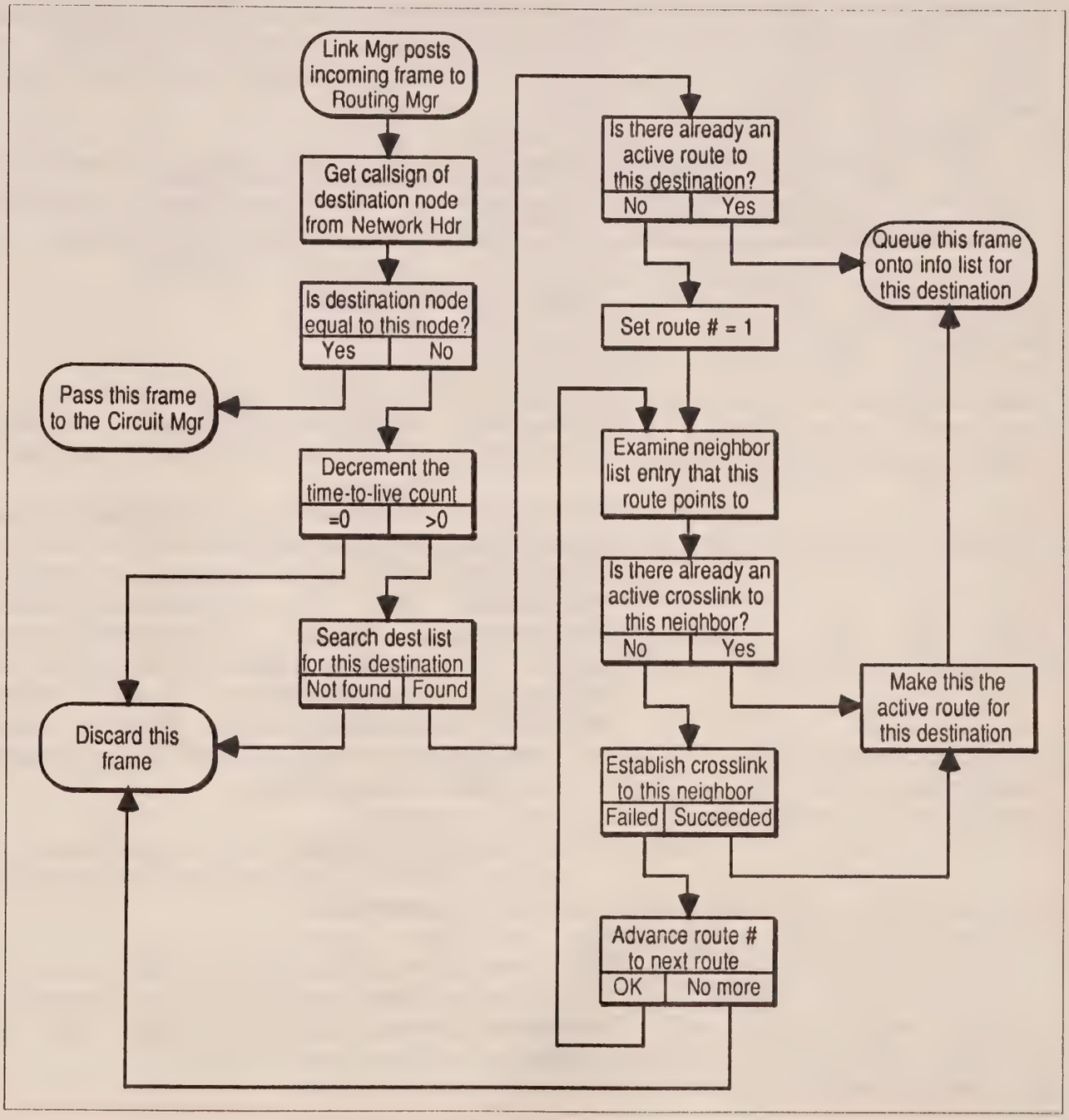
For each node in the destination list, the routing table up to three *routes* to that destination node. In this context, a route simply identifies a neighboring node that is a step closer to the ultimate destination. For each route, the destination list maintains a *quality* value which quantifies the relative desirability of each route. NET/ROM maintains route quality as an integer which ranges from 255 (best) to 0 (worst). Routes are maintained in sorted order by quality, and NET/ROM always attempts to use the highest-quality route available. It also keeps an *obsolescence count* , which enables NET/ROM to purge paths from its routing table when it has become unuseable and remained so for a protracted period of time.

Observe that the routing table does *not* contain the *entire path* to each known destination (this is unnecessary, and would require too much memory), but just a list of neighboring nodes that are reasonable choices for a next step enroute to the destination. You can ask to see the routes to a particular destination node by using a NODES command that specifies the callsign or mnemonic identifier of the destination.

Appendix B — Routing

Routing Algorithm

The Routing Manager analyzes the network header of each incoming 'CF'-frame, and determines how to route that frame. The network header contains the callsign of the destination node and a "time-to-live" counter. (For the layout of the network header, see the diagram on the preceding page.) The routing algorithm is straightforward, and is described in the following flow diagram:

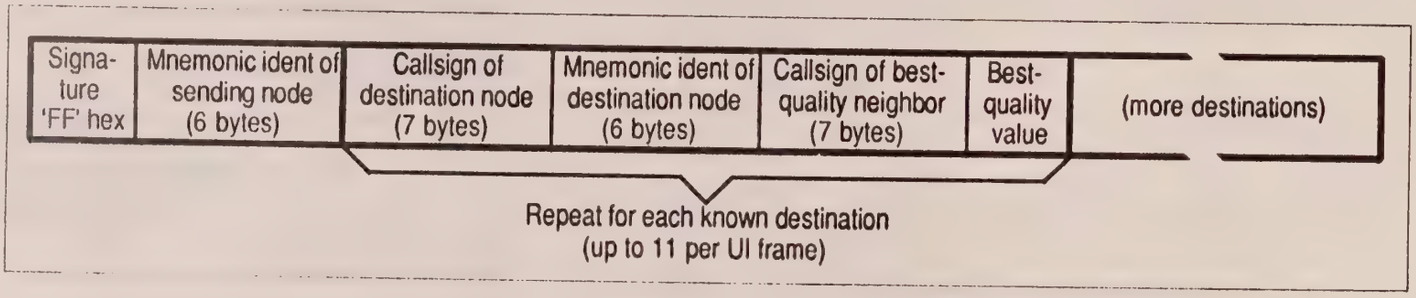

Link Mgr posts incoming frame to Routing Mgr
Get callsign of destination node from Network Hdr
Is destination node equal to this node?
Yes
No
Pass this frame to the Circuit Mgr
Decrement the time-to-live count
=0
>0
Search dest list for this destination
Not found
Found
Discard this frame
Is there already an active route to this destination?
No
Yes
Queue this frame onto info list for this destination
Set route # = 1
Examine neighbor list entry that this route points to
Is there already an active crosslink to this neighbor?
No
Yes
Make this the active route for this destination
Establish crosslink to this neighbor
Failed
Succeeded
Advance route # to next route
OK
No more

Appendix B — Routing

Automatic Routing Table Updates

Each node makes a periodic broadcast of information from its routing table in order to provide the basis of routing table updates at neighboring nodes. The broadcast is normally made once an hour, although the frequency may be changed with the PARMS command by the control operator.

The routing broadcast takes the form of one or more AX.25 UI-frames tagged with PID='CF'. The source callsign identifies the broadcasting node, and the destination callsign is "NODES". The information contents of each frame has the following structure:


Signature 'FF' hex
Mnemonic ident of sending node (6 bytes)
Callsign of destination node (7 bytes)
Mnemonic ident of destination node (6 bytes)
Callsign of best-quality neighbor (7 bytes)
Best-quality value
(more destinations)
Repeat for each known destination (up to 11 per UI frame)

Up to 11 destinations are packed into each UI-frame, and the node broadcasts as many such frames as required to send its entire routing table.

When a node receives an auto-routing broadcast UI-frame from one of its neighboring nodes, it analyzes the contents and makes appropriate updates to its own routing table. This process is more complex than one might think. The receiving node utilizes a series of heuristic rules to keep the size of its routing table manageable and to try to avoid "loops" and other undesirable routes. Here is a summary of the most important rules used in processing auto-routing update broadcasts:

❶ If the *worst quality for auto-updates* parameter (set by PARMS) is zero, all auto-update broadcasts are ignored.

❷ If the if the PID of the UI-frame is not 'CF' hex, or if the first byte of its contents is not the proper *signature* ('FF' hex), the frame is ignored.

Appendix B — Routing

Automatic Routing Table Updates (continued)

❸ A direct route is assumed to exist to the node that originated the broadcast. The quality of this route is set in accordance with the *channel quality* parameter (set by PARMS) appropriate to the channel (HDLC or RS232) over which the broadcast was received.

❹ For each destination node listed in the broadcast, an indirect route is assumed to exist via the node who originated the broadcast. The quality of this route is calculated by combining the broadcast quality with the channel quality parameter appropriate to the channel over which the broadcast was received. The qualities are multiplied, normalized, and rounded as shown in the following formula:

$$\text{routequality} = \frac{(\text{broadcastquality} \times \text{channelquality}) + 128}{256}$$

❺ An indirect route is considered to be a *trivial loop* if the callsign of the best-route neighbor node in the broadcast matches the callsign of the receiving node. Trivial loop routes are assigned a quality value of zero, so they are used only as a last resort. Quality-zero routes are never included in outgoing auto-routing broadcasts.

❻ Only the three higest-quality routes to a destination are retained.

❼ Any route with quality less than the *worst quality for auto-updates* parameter (set by PARMS) is ignored.

❽ If the number of entries in the destination list is greater than or equal to the *max destinations* parameter (set by PARMS), then no additional destinations will be added.

Each route in the routing table has an *obsolescence count* which is initialized to the "obsolescence count initializer" value (set by PARMS) whenever the route is added or updated as the result of an auto-routing broadcast. The count is also reinitialized to this value whenever the route is actually used successfully. The default initializer value is 6.

Automatic Routing Table Updates (continued)

Periodically, NET/ROM scans through the routing table and decrements the obsolescence count of every route in the table—this scan occurs with the same frequency as routing broadcasts, normally once each hour. The obsolescence count of a route is also decremented whenever the route fails in actual use. If the obsolescence count of a route is decremented to zero, the route is deleted from the routing table.

A routing table entry created by the automatic routing update mechanism can never have an obsolescence count of zero, since such an entry is automatically purged from the table when its count reaches zero. When a route is entered into the routing table manually with the NODES+ command, however, it *is* possible to set the route's obsolescence count to zero. This has special significance: it marks the route as *permanent*. Such a permanent route will never be updated or deleted by the auto-update mechanism. It can, however, be deleted manually with the NODES- command.

Appendix C — Parameters

This section describes in detail the function of the 24 NET/ROM parameters that the control operator can change remotely using the PARMS command.

Parameter 1: Maximum destination list entries

(default=50, minimum=1, maximum=400)
Defines the maximum allowable number of destinations in the node's routing table. Each destination consumes 32 bytes of RAM. The control operator can use this parameter to limit the amount of RAM that is allocated to the routing table, thus ensuring that sufficient space remains for frame buffering.

Parameter 2: Worst quality for auto-updates

(default=1, minimum=0, maximum=255)
Defines the poorest route quality that will be automatically added to the node's routing table. The control operator can use this parameter to limit the automatic routing update function to accept only higher-quality routes. In addition, the automatic update function can be disabled altogether by setting this parameter to zero.

Parameter 3: Channel 0 (HDLC) quality

(default=192, minimum=0, maximum=255)
Defines the quality of the radio channel connected to the node's HDLC port. The control operator should set this parameter to an appropriate quality value in accordance with the speed, reliability, and congestion anticipated on the channel. The default value of 192 is appropriate for a 1200-baud user-accessible frequency...if the actual channel quality is better (e.g., a UHF backbone frequency) or worse (e.g., an HF link), the parameter value should be changed accordingly.

Appendix C — Parameters

Parameter 4: Channel 1 (RS232) quality

(default=255, minimum=0, maximum=255)
Defines the quality of the TNC-to-TNC interconnect channel connected to the node's RS232 port. The control operator should set this parameter to an appropriate quality value in accordance with the speed, reliability, and congestion anticipated on the channel. The default value of 255 is appropriate for a 9600-baud two-modem interconnect cable...if the actual channel quality is worse (e.g., a three- or four-port interconnect, or a satellite link), the parameter value should be changed accordingly.

Parameter 5: Obsolescence count initializer

(default=6, minimum=0, maximum=255)
Defines the initial value given to the obsolescence count of a route that has been newly added or updated by the node's automatic routing table update mechanism. The obsolescence count of a route is also reinitialized to this value whenever the route is actually used successfully. The obsolescence count of a route is decremented once each auto-update broadcast interval. However, such periodic decrementing of route obsolescence counts can be disabled altogether by setting this parameter to zero.

Parameter 6: Obsolescence count minimum to be broadcast

(default=5, minimum=1, maximum=255)
Defines the minimum obsolescence count threshhold below which a route will not be included in the node's automatic routing broadcasts. The purpose of this threshhold is to prevent the node from broadcasting "stale" routing information to other nodes. Under normal circumstances, this parameter should be assigned a value no greater than the value of parameter 5 (*obsolescence count initializer*); if it is greater, the node's broadcasts will include no destinations other than itself.

Appendix C — Parameters

Parameter 7: Auto-update broadcast interval (seconds)

(default=3600, minimum=0, maximum=65535)
Defines the number of seconds between automatic routing broadcasts issued by the node. The default value of 3600 specifies an hourly broadcast. In addition, broadcasts can be disabled altogether by setting this parameter to zero.

Parameter 8: Network "time-to-live" initializer

(default=64, minimum=0, maximum=255)
Defines the initial value of the "time-to-live" field in the Network Header of all network-layer frames originated by this node. The time-to-live field is decremented by each intermediate node that relays the frame. If the time-to-live value ever reaches zero, the frame is discarded. This protects the network against frames persisting forever as the result of a routing loop. The value of this parameter should be a bit larger than number of "hops" in the longest legitimate route in the network.

Parameter 9: Transport timeout (seconds)

(default=60, minimum=5, maximum=600)
Defines the number of seconds between transport-layer retries.

Parameter 10: Transport maximum tries

(default=3, minimum=2, maximum=127)
Defines the maximum number of transport-layer tries attempted before a circuit failure is reported.

Parameter 11: Transport acknowledge delay (seconds)

(default=3, minimum=1, maximum=60)
Defines the number of seconds' delay used by the transport layer from the time it receives an information message until it sends an information-acknowledge message. The purpose of this delay is to give the acknowledgement an opportunity to be "piggybacked" upon an outgoing information message.

Appendix C — Parameters

Parameter 12: Transport busy delay (seconds)

(default=180, minimum=1, maximum=1000)
Defines the maximum number of seconds that the transport layer will remain "choked" as the result of an incoming message that has the choke flag bit set. The purpose of this timeout is to prevent an indefinite hangup in the event that the "unchoke" message is lost.

Parameter 13: Transport requested window size (frames)

(default=4, minimum=1, maximum=127)
Defines the maximum number of incoming out-of-sequence information messages that the transport layer will buffer while waiting for the next expected information message to arrive. Also defines the maximum number of outgoing information messages that the transport layer will send without receiving acknowledgement.

Parameter 14: Congestion control threshhold (frames)

(default=4, minimum=1, maximum=127)
Defines the maximum allowable backlog of messages that the transport layer will buffer before it sends a choke message. Also defines the maximum allowable backlog of frames that the link layer will buffer before it sends an RNR control frame.

Parameter 15: No-activity timeout (seconds)

(default=900, minimum=0, maximum=65535)
Defines the maximum number of seconds that a transport-layer circuit or a link-layer connection can remain idle (i.e., no information transfer in either direction) before it is automatically disconnected.

Parameter 16: Link digipeater wait "DWAIT" (10ms increments)

(default=16, minimum=0, maximum=127)
Defines the extra clear-channel interval (measured in 10-millisecond increments) that the link-layer requires before keying up the transmitter to send frames originated by the node. (This extra interval is not required to send digipeated frames.) For networks in which digipeating is seldom used, the control operator may wish to set this parameter to zero.

Appendix C — Parameters

Parameter 17: Link T1 timeout "FRACK" (seconds)

(default=4, minimum=1, maximum=15)
Defines the number of seconds between link-layer retries. When digipeating is used, this value is multiplied by *2D+1*, where *D* is the number of digipeaters.

Parameter 18: Link transmit window size "MAXFRAME" (frames)

(default=7, minimum=1, maximum=7)
Defines the maximum number of outgoing information frames that the link layer will send without receiving acknowledgement.

Parameter 19: Link maximum tries

(default=10, minimum=0, maximum=127)
Defines the maximum number of tries that the link layer will attempt before reporting a link failure. If this parameter is set to zero, the link layer will retry forever (*not recommended*).

Parameter 20: Link T2 timeout (10ms increments)

(default=100, minimum=0, maximum=65535)
Defines the delay (measured in 10-millisecond increments) used by the link layer from the time it receives an information frame until it sends an acknowledgement (RR, RNR, or REJ) control frame. The purpose of this delay is to give the acknowledgement an opportunity to be "piggy-backed" upon an outgoing information frame.

Parameter 21: Link T3 timeout (10ms increments)

(default=18000, minimum=0, maximum=65535)
Defines the maximum no-activity period (measured in 10-millisecond increments) permitted by the link layer before it issues a poll to make sure the link is still intact. This timeout is also used to break link-layer choke deadlocks.

Parameter 22: AX.25 digipeating (1=enabled, 0=disabled)

(default=1, minimum=0, maximum=1)
Defines whether or not the node will perform AX.25 digipeating. The default value of 1 causes digipeating to be enabled.

Parameter 23: Validate callsigns (1=enabled, 0=disabled)

(default=1, minimum=0, maximum=1)
Defines whether or not the node will perform validation checks on amateur callsigns. The default value of 1 causes callsign validation to be enabled.

Parameter 24: Station ID beacons (1=enabled, 0=disabled)

(default=1, minimum=0, maximum=1)
Defines whether or not the node will broadcast station-identification beacons every 9 minutes and 59 seconds. The default value of 1 causes this feature to be enabled.

Appendix D — Operating Notes

Fixed Limits

The following fixed limits apply to the current version of NET/ROM:

- Maximum number of links per node: 25
- Maximum number of circuits per node: 20

Watchdog Timers

Some digipeater operators have been known to incorporate a "watchdog timer" circuit to prevent the possibility of a TNC hangup due to software failure. The simplest approach is to rig up a 555 timer that resets the TNC every 10 or 15 minutes.

Trying to use such a "brute force" approach on a NET/ROM node would be disasterous. Reseting a NET/ROM node causes instant termination of all uplinks, downlinks, and crosslinks to or from that node. *Don't even consider it!*

A somewhat more sophisticated approach is to set up a retriggerable timer triggered by the TNC's push-to-talk output. The idea is to reset the TNC only if PTT has not been activated for some period of time. This approach will probably work OK with NET/ROM as long as the timeout period is at least 15 minutes or so. The automatic station-identification broadcasts every 9 minutes and 59 seconds should prevent such a timer from firing unless something goes terribly wrong. If you use this approach, *don't turn off the station-identification feature!*

Our experience is that the hardware and firmware are extremely reliable, and consequently this sort of timer is rarely necessary.

Appendix D — Operating Notes

Long Frames

To achieve best efficiency using NET/ROM, users may wish to set their TNC's maximum frame length parameter ("PACLEN") to 236 bytes or less. The maximum allowable AX.25 information frame is 256 bytes. Since NET/ROM inter-node protocols impose an additional overhead of 20 bytes, the transport-layer information field is limited to a maximum of 236 bytes. User frames longer than 236 bytes are automatically fragmented into two transport frames by the originating node, and reassembled into one long frame again by the destination node. This fragmentation and reassembly is completely transparent to users of the network, so the only disadvantage of sending long frames (>236 bytes) is some additional overhead.

Callsign Validation

NET/ROM includes a callsign validation algorithm whose purpose is to protect the network against invalid amateur callsigns in commands, link headers, routing table entries, etc. To qualify as a valid amateur callsign, a callsign must meet the following criteria:

❶ Length of callsign (excluding SSID) must be between 4 and 6 characters.
❷ All characters must be letters or digits (no punctuation or control characters).
❸ The callsign must contain either 1 or 2 digits.
❹ The last character of the callsign must be a letter (not a digit).
❺ The SSID suffix (if present) must be in the range 0 to 15.

In practice, one of the most useful functions of such callsign validation is to prevent uplinks from improperly-initialized user TNCs which are using a bad callsign (most commonly, "NOCALL", "PK64", or blank).

The PARMS command includes a parameter that allows a control operator to defeat this callsign validation if desired. *However, we strongly advise against disabling callsign validation in NET/ROM, and we disclaim any responsibility for problems that occur in nodes operating with callsign validation disabled.*

Appendix D — Operating Notes

Multi-Connect Operation

Since many TNCs now support multiple AX.25 connections, you may wish to establish multiple simultaneous uplinks to your local NET/ROM node. This can be done successfully, but some precautions are in order. The AX.25 protocol is not capable of supporting multiple simultaneous connections in which both originator and destination callsigns are identical. Consequently, to achieve multiple concurrent uplinks from your TNC to a node, you must employ one of the following tactics:

Tactic one ☞

If your TNC is capable of supporting independent originator callsigns ("MYCALLs") on each channel, then you can establish different callsigns for each channel (typically, your call with SSIDs -0, -1, -2, etc.) and then uplink with impunity. Since each uplink will have a unique originator callsign, there will be no protocol ambiguity. This is the best approach...but unfortunately, many TNCs are incapable of assigning independent originator callsigns to each channel.

Tactic two ☞

If your local NET/ROM node (let's say it's "W6AMT-3") has a mnemonic identifier (let's say it's "LAX"), then the node will actually respond to seventeen different names: its callsign (W6AMT-3) plus its identifier with any valid SSID (LAX, LAX-1, LAX-2,..., LAX-15). Consequently, you can establish multiple concurrent uplinks to the node as long as you are careful to call the node by a different name when establishing each uplink.

Appendix D — Operating Notes

Assigning Channel Quality Values

For both channel 0 (HDLC) and channel 1 (RS232), NET/ROM maintains channel quality parameters which are set by PARMS and used in route quality calculations. The quality of a route segment is a value from 0 to 255 (255 is best, 0 is worst), and the quality of a route is the normalized product of the quality of its segments.

Channel quality depends upon baud-rate, congestion, and many other factors. It is up to the NET/ROM control operators to assign reasonable quality values to the various available channels in order to allow NET/ROM's optimum routing algorithm to make the most efficient routing choices. As a starting point, we suggest the following values to be assigned to channel quality parameters:

Channel Description	Quality	% Perfect
9600-baud RS232 wire interconnect (2-port)	255	99%+
9600-baud RS232 satellite interconnect (2-port)	252	98%
9600-baud RS232 wire interconnect (3-port)	248	97%
9600-baud HDLC isolated internode backbone	240	94%
1200-baud HDLC isolated internode backbone	224	88%
1200-baud HDLC user-accessed channel	192	75%
300-baud HDLC HF channel	128	50%
Known loopback or route of unknown quality	0	?

The default channel quality values are quality=192 for channel 0 (HDLC), and quality=255 for channel 1 (RS232). For nodes connected to high-quality backbone channels, you should use PARMS to change the channel 0 quality to an appropriate value (as illustrated in the table above). Likewise, for multi-channel nodes with satellite interconnects or with 3 or more ports, your should change the channel 1 quality.

The worst possible quality value (quality=0) is reserved by NET/ROM to indicate a trivial loop or route of unknown quality that should be used only as a last resort. Quality 0 paths are never propagated to other nodes via auto-routing broadcasts. In most situations, you should not assign a value of 0 to either of the channel quality parameters.

Appendix D — Operating Notes

Overriding NET/ROM's Automatic Routing Mechanisms

NET/ROM is designed to create and maintain its routing tables on a fully automatic basis, without requiring manual intervention. There may be occasions, however, when you as a control operator wish to "take matters into your own hands" by overriding NET/ROM's automatic routing mechanisms. This can be done is several ways.

You can use the NODES- command to delete automatically-created entries from the routing table, and the NODES+ command to create new entries manually or to modify existing entries. When you enter a route manually using NODES+, one of the parameters you must provide is the initial value of the route's obsolescence count. This value determines how long the route will remain in the routing table before being automatically deleted. If you set the obsolescence count to zero with the NODES+ command, then the route is "locked", and will never be deleted or modified by NET/ROM. Such a locked entry will remain in the routing table unchanged until manually deleted with NODES-.

You can also disable various portions of NET/ROM's automatic routing mechanism by using the PARMS command. Setting parameter #2 (*worst quality for auto-updates*) to zero causes NET/ROM to ignore incoming auto-routing broadcasts. Setting parameter #5 (*obsolescence count initializer*) to zero inhibits the automatic deletion of obsolete routes. Setting parameter #6 (*obsolescence count minimum to be broadcast*) to 255 inhibits broadcast of auto-routing information for destinations other than the node itself. Setting parameter #7 (*auto-update broadcast interval*) to zero inhibits outgoing auto-routing broadcasts altogether. So, for example, you can disable all automatic routing functions by zeroing parameters #2 and #7.

We suggest, however, that you think very carefully before disabling any of the automatic routing functions of a node. Under normal circumstances, the automatic routing mechanisms of NET/ROM do an excellent job of keeping routing table information up-to-date. Furthermore, each NET/ROM node has a "responsibility" to the other nodes in

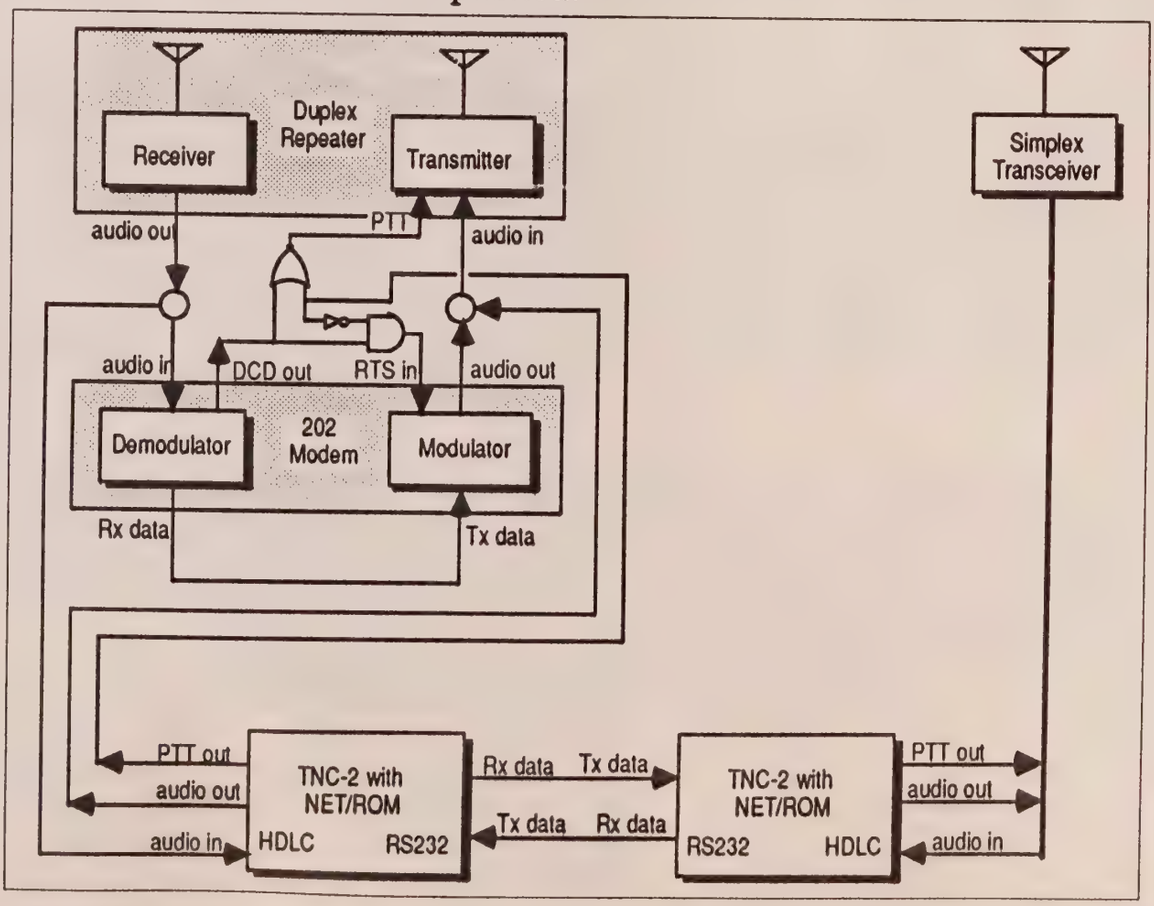
Appendix D — Operating Notes

Overriding NET/ROM's Automatic Routing Mechanisms (continued)

the network to process and broadcast automatic routing information. Disabling the automatic routing functions of one node may affect the routing tables of many other nodes. Therefore, you should probably consider disabling NET/ROM's automatic routing function only under extraordinary circumstances—for example, to protect your node and the rest of the network against "pollution" from a malfunctioning node broadcasting corrupted routing information.

Interfacing with a Duplex Repeater

In high-density areas with lots of congestion on packet user frequencies, a duplex packet repeater can improve throughput and reduce collisions significantly in comparison to an ordinary simplex digipeater. A dual-port NET/ROM node can be readily interfaced with such a repeater in such a way as to give users on the duplex repeater access to the rest of the normal simplex network. The following diagram illustrates one way in which this can be accomplished.



Appendix E — Index

A

Aborting a CONNECT command, 36
Adaptive routing, 28
Adopting the user's callsign, 13
Algorithm, routing, 63
Alias, 33, 35
Alternate routes, 5, 8, 28, 62
Authors, 1
Auto-forwarding BBSs, 33
Automatic routing, 28
Automatic routing
  broadcasts, 31, 64
  overriding, 77
  updates, 8, 31, 64

B

Backbone, 15
Baud-rate switches, 49
BBS forwarding, 33
Broadcast
  automatic routing, 31, 64, 77
  station identification, 33
Busch, Michael D. [W6IXU], 1

C

Callsign
  hard-coded into EPROM, 50
  validation, 74
  verifying node's, 51
Channel quality, 32, 65, 76
Circuit, 5, 12, 21, 39
Circuit manager, 21, 24-25
Command interpreter, 22-26
Command interpreter messages, 43
Commands, 9, 21-22, 35
Commands
  CONNECT, 13-14, 26, 28, 35
  IDENT, 40-41
  NODES, 20, 28, 30-31, 34, 37-38, 40, 62
  NODES+, 38, 66, 77
Commands
  NODES-, 38, 66, 77
  PARMS, 34, 40, 42, 67
  RESET, 34, 40, 43
  SYSOP, 22-23, 31, 34, 38, 40-41, 43, 45
  USERS, 39
CONNECT, 9, 13-14, 22, 26, 28, 35
Connect acknowledge (L4), 56
Connect host (ESC-C), 45
Connect request (L4), 56
Control operator, 34, 40
Copyright, 1
Coupler, diode-matrix, 5, 52
CPU clock speed, 48
Crosslink, 12, 14, 25

D

Data switch, 21, 24-26
Datagram network service, 5
DCD-B wired to RS232 connector, 49
Deferred disconnect, 7, 27
Destination list, 28-29, 61
Digipeater, 11
Digipeating, 5-6, 9, 17, 33, 35
Digipeating vs. store-and-forward, 17
Disconnect acknowledge (L4), 57
Disconnect host (ESC-D), 45
Disconnect request (L4), 57
Downlink, 11, 13-14, 26, 35, 39
Dual-channel node, 5, 7, 11, 15, 52
Duplex repeater, 78

E

End-to-end transport protocol, 55
EPROM, 6, 50
Error control, 6, 18, 21, 27
ESC-C (host connect), 45
ESC-D (host disconnect), 45
ESC-P (host password string), 45
ESC-T (host TXDELAY), 46
ESC-Y (host enable/disable), 46

Appendix E — Index

F

Firmware organization, 19
Fixed limits, 73
Flow control, 6, 21, 27
Flow of information, 24
Frames longer than 236 bytes, 74

H

Hardware setup, 47
HDLC frame structure, 55
Hidden nodes, 37, 41
Host
  enable/disable (ESC-Y), 46
  interface, 23, 39, 45
  interface messages, 46
  terminal, 36, 45, 50
  terminal cable, 50

I

IDENT, 9, 22, 34, 40-41
Identifier (mnemonic), 8, 35, 41, 51
Information acknowledge (L4), 58
Information flow, 24
Information message (L4), 57
Installation of NET/ROM, 47-54
Inter-node protocol, 55
Interconnect
  protocol (AX.25 variant), 59
  TNC-to-TNC cable, 7, 52
ISO reference model, 19

L

Limits (fixed), 73
Link, 11
Link manager, 20, 24-26
Local updates to routing table, 31, 38
Locked route entries, 38, 66
Long frames (> 236 bytes), 74
Loops (trivial), 65

M

Manual routing updates, 8, 38, 77
Messages
  command interpreter, 43
  host interface, 46
Mnemonic identifier, 8, 35, 41, 51
Multi-channel node, 5, 7, 9, 11, 16, 52
Multi-connect operation, 75
Multi-port digipeating, 33

N

Neighbor list, 28-29, 61
Network header, 55
Networking 5
Node, 5, 11, 35
NODES, 9, 20, 22, 28, 30-31, 34, 37-38, 40, 62
NODES+, 38, 66, 77
NODES-, 38, 66, 77

O

Obsolescence count, 30, 37-38, 62, 65
Op-amp (upgrade U3), 49
Organization of the firmware, 19
Overriding automatic routing, 77

P

PACLEN, 74
Parameter setup, 51
Parameters (PARMS), 42, 67-72
PARMS, 9, 22, 34, 40, 42, 67-72
Password string, 23, 34, 40, 45, 51
Patchcord, 21, 25, 39
Permanent route entries, 38, 66, 77
Piggyback acknowledgement (L4), 57
Protocol ID (PID), 20, 31, 63-64
Protocols, 5, 55-60
PTT failsafe timer, 48

Appendix E — Index

Q

Quality
  channel, 32, 65, 76
  route, 28, 30, 32, 37-38, 62

R

Raikes, Ronald E. [WA8DED], 1
RAM
  free space displayed by USERS, 39
  upgrade from 16K to 32K, 47
Remote routing updates, 31, 38
Repeater (duplex), 78
RESET, 9, 22, 34, 40, 43
Route quality, 28, 30, 32, 37-28, 62, 65
Routing, 5, 8, 28
Routing
  algorithm, 63
  automatic adaptive, 8
  broadcast, 31
  manager, 20, 25, 28
  table, 25, 28, 61
  table structure, 61
  table updates, 8, 31
RS232 interconnect protocol, 59

S

Satellite links, 7
Security, 34
Serial drivers, 19
Sliding window protocol, 6, 21, 27, 55
Software 2000, Inc., 1
SSID, 14, 33
Station identification, 33, 73
Store-and-forward, 5-6, 18
SYSOP, 9, 22-23, 31, 34, 38, 40-41, 43, 45

T

Terminology, 11
Time-constant of PTT failsafe, 48
Time-to-live count, 63
Timer
  PTT failsafe, 48
  watchdog, 73
TNC-2, 5-6, 47
Trademark, 1
Transport header, 55
Transport layer
  circuit, 12
  protocol, 55-60
Transport service, 5
Trivial loop, 65
TXDELAY (ESC-T), 46

U

Unattended operation, 45-46, 51
Upgrading RAM to 32K, 47
Uplink, 11, 13-14, 24, 39
Usage example, 9
User, 11
USERS, 9, 22, 39

V

Validation
  of control operator, 34, 40
  of callsigns, 74
Virtual circuit, 5, 12

W

W0RLI BBS, 33
W6IXU, 1
WA8DED, 1
Warm-start, 43
Warranty, 1
Watchdog timer, 73
Wormhole, 7

Appendix E — Index

(Intentionally left blank.)